Building a Literate Parser and Proxy for DNP3

Sven M. Hallberg, Sergey Bratus, Adam Crain

TUHH
Technische Universität Hamburg-Harburg

DARTMOUTH
VOX CLAMANTIS IN DESERTO
1769

AUTOMATAK

SEL
SCHWEITZER ENGINEERING LABORATORIES, INC.

uh

Outline

- Parsers, security, and the LangSec viewpoint
- Building a safer DNP3 parser from scratch

  “Make the parser code look like the grammar”

  A.k.a. *Parser combinators* (using the Hammer kit from UpstandingHackers.com)
- Case study: a DNP3 filtering proxy

  Validating (testing) our implementation
- Lessons learned / discussion

LangSec

- Many security issues are **language recognition** issues
  exploit = accepting bad input, letting it act on program
  internals. What to accept? What is expected? What is valid?

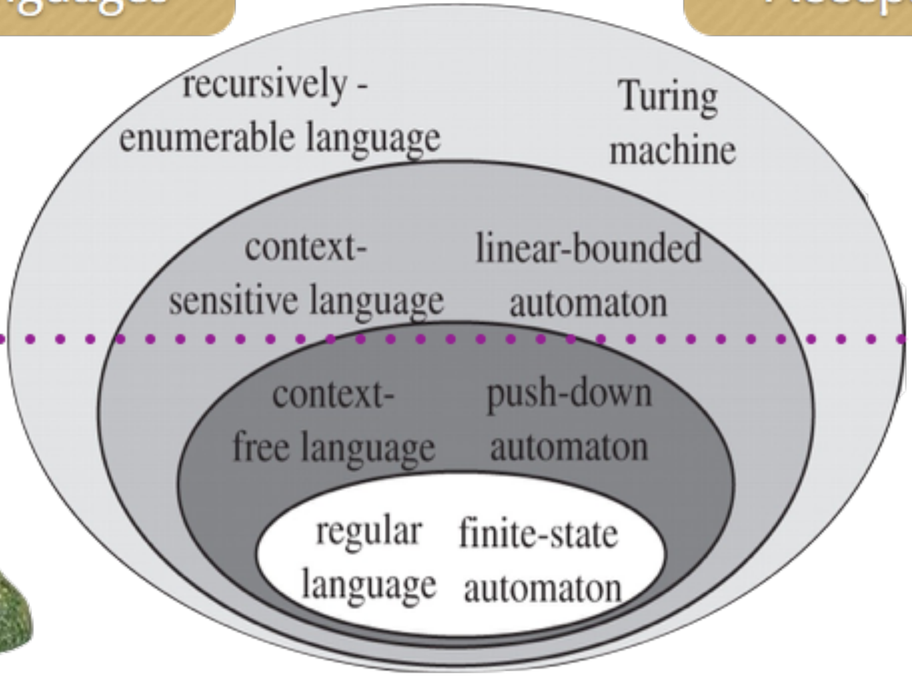
- If security seems like an uphill battle…
  Just look at the syntax complexities. (there's a theory of it:
  Chomsky hierarchy of grammars)

- Some syntax is poison: (eg.: nested length, fields that must all agree; several sources of truth, …)

Languages vs recognizers

Languages

Acceptors



"Here be dragons"

"The Shire of validation"

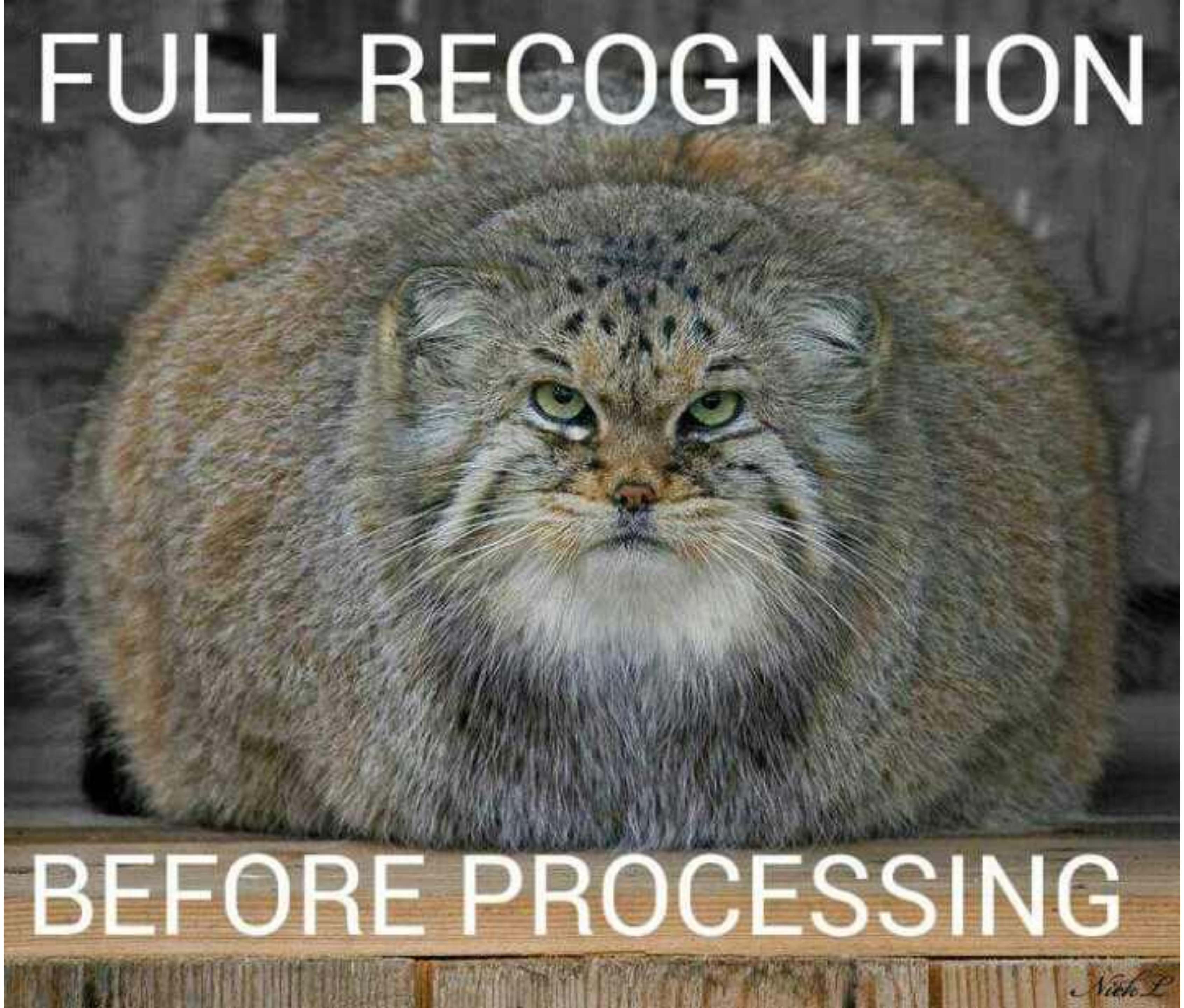
Solve language problems with a language approach

- Start with a grammar
  - If you don’t know what valid or expected syntax/content of a message is, how can you check it? Or interoperate?
  - If the protocol comes without a grammar, you need to derive one. It sucks, but it’s the only way.

- Write the parser to look like the grammar: succinct, *incrementally testable* (from the leaf nodes/primitives up)

- Don’t start processing before you’re done parsing

FULL RECOGNITION
BEFORE PROCESSING


FULL RECOGNITION
BEFORE PROCESSING
Nick L

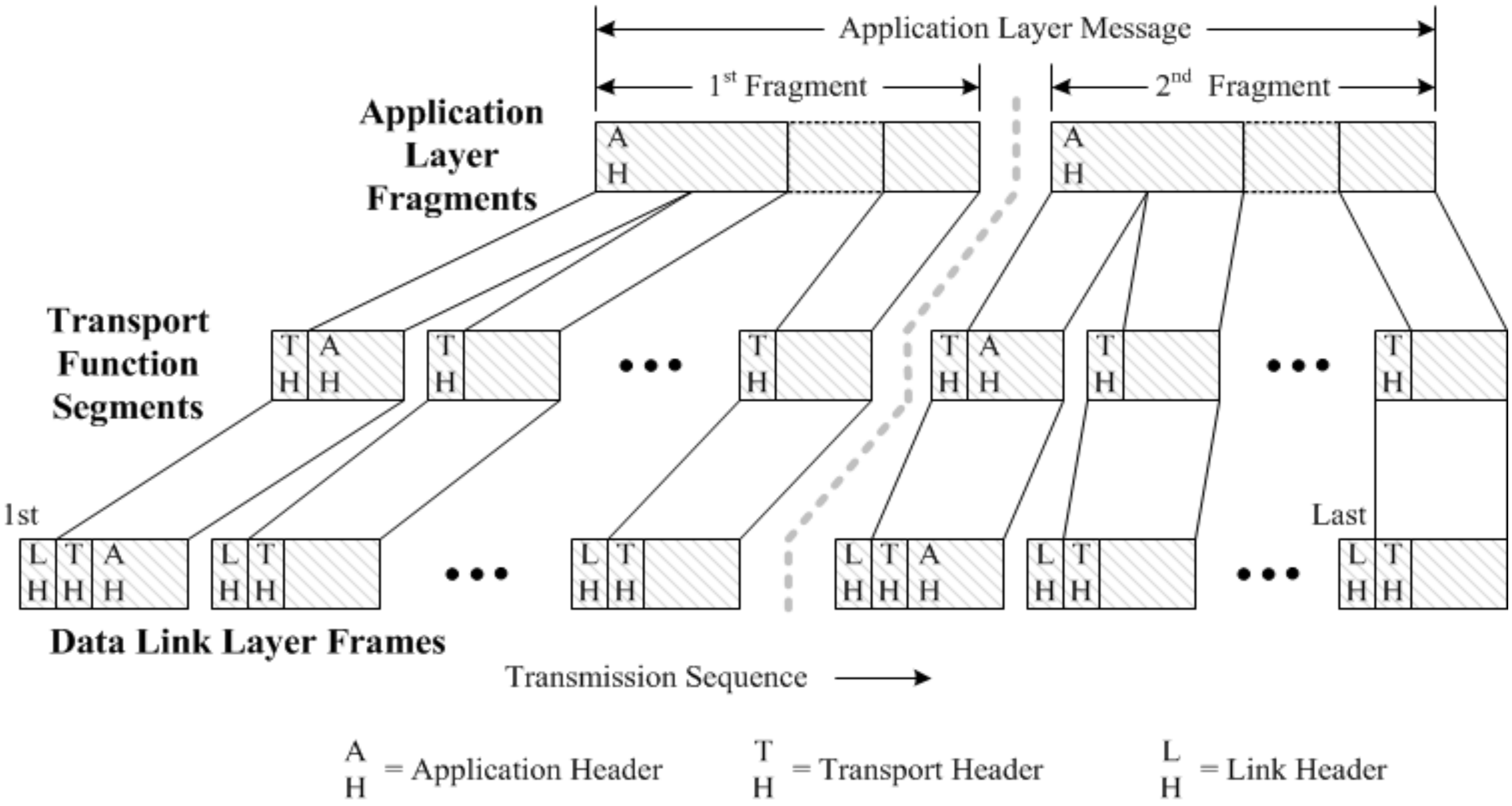
DNP3 issues are not theoretical

- 2013 to 2014 – Over 30 CVEs related to input validation with DNP3 implementations.

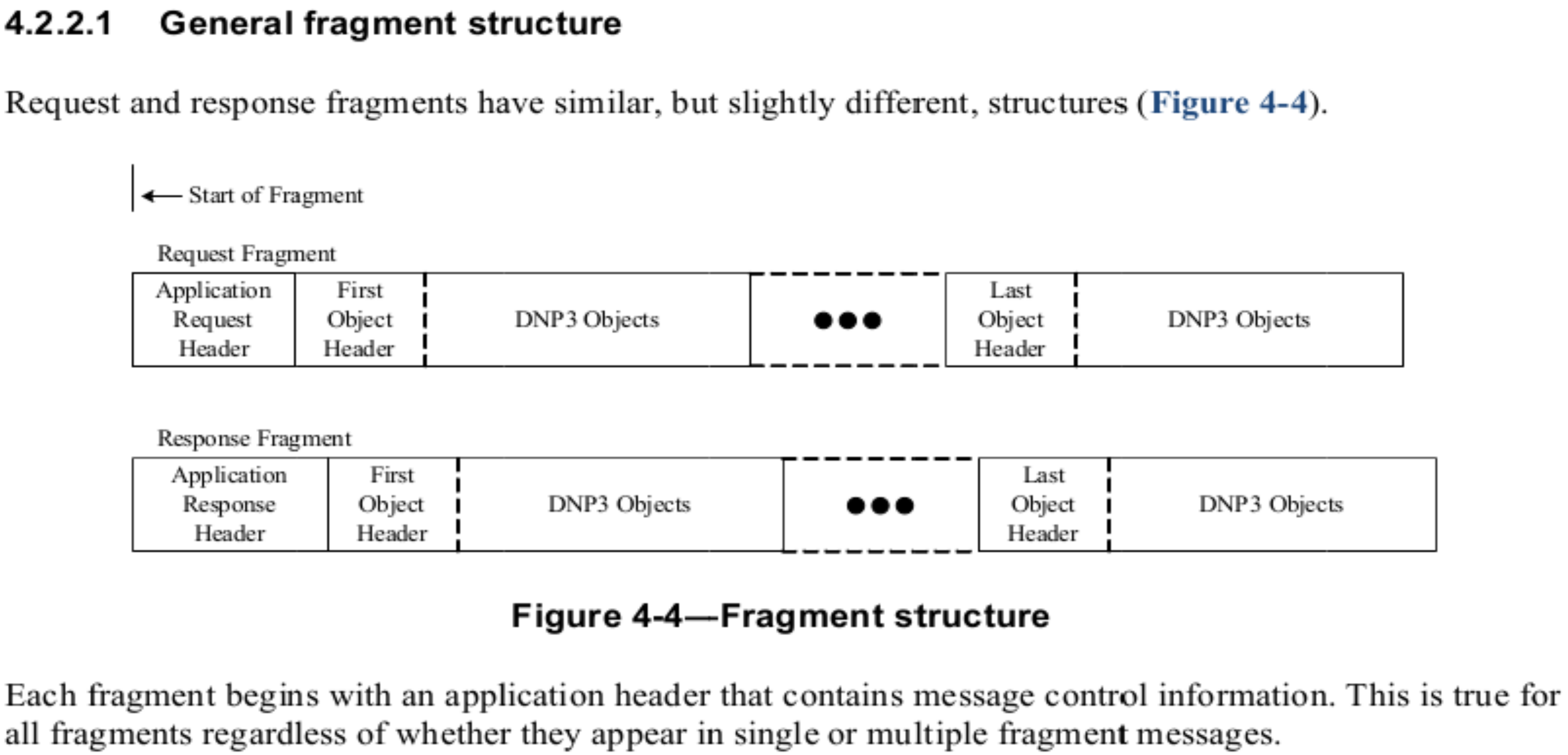
- Out of dozens of implementations only a small few were defect-free.

- Low-defect implementations chose a conservative subset

DNP3 Complex?



DNP3 Complex??



4.2.2.1 General fragment structure

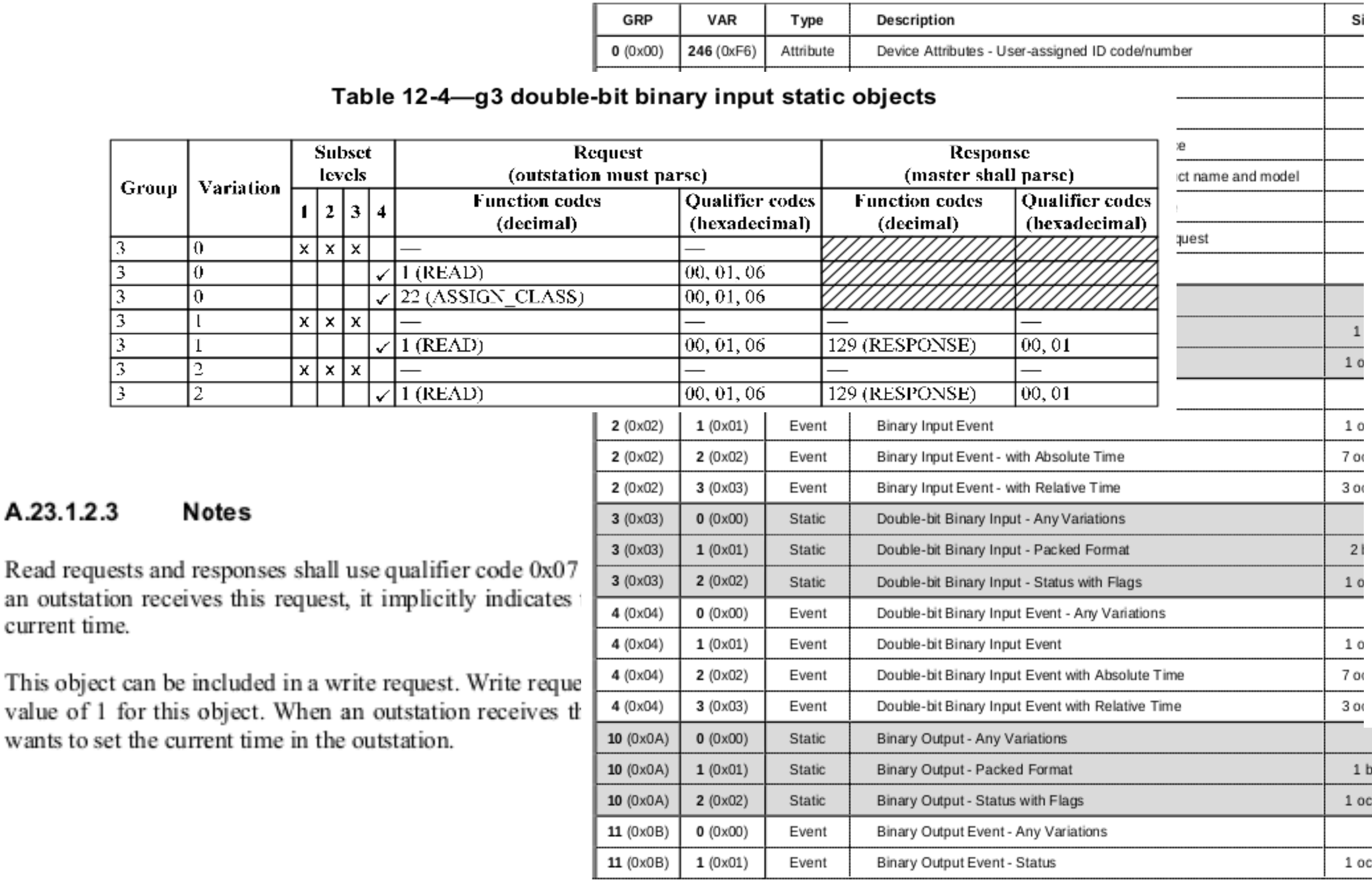
Request and response fragments have similar, but slightly different, structures (Figure 4-4).

← Start of Fragment
Request Fragment
Application Request Header
First Object Header
DNP3 Objects
●●●
Last Object Header
DNP3 Objects
Response Fragment
Application Response Header
First Object Header
DNP3 Objects
●●●
Last Object Header
DNP3 Objects

Figure 4-4—Fragment structure

Each fragment begins with an application header that contains message control information. This is true for all fragments regardless of whether they appear in single or multiple fragment messages.

DNP3 Complex!?!



GRP	VAR	Type	Description	Si		
0 (0x00)	246 (0xF6)	Attribute	Device Attributes - User-assigned ID code/number			
			e			
			ct name and model			
			quest			
				1		
				1 o		
2 (0x02)	1 (0x01)	Event	Binary Input Event	1 o		
2 (0x02)	2 (0x02)	Event	Binary Input Event - with Absolute Time	7 o		
2 (0x02)	3 (0x03)	Event	Binary Input Event - with Relative Time	3 o		
3 (0x03)	0 (0x00)	Static	Double-bit Binary Input - Any Variations			
3 (0x03)	1 (0x01)	Static	Double-bit Binary Input - Packed Format	2 I		
3 (0x03)	2 (0x02)	Static	Double-bit Binary Input - Status with Flags	1 o		
4 (0x04)	0 (0x00)	Event	Double-bit Binary Input Event - Any Variations			
4 (0x04)	1 (0x01)	Event	Double-bit Binary Input Event	1 o		
4 (0x04)	2 (0x02)	Event	Double-bit Binary Input Event with Absolute Time	7 o		
4 (0x04)	3 (0x03)	Event	Double-bit Binary Input Event with Relative Time	3 o		
10 (0x0A)	0 (0x00)	Static	Binary Output - Any Variations		1, 22	
10 (0x0A)	1 (0x01)	Static	Binary Output - Packed Format	1 bit	1, 2	129
10 (0x0A)	2 (0x02)	Static	Binary Output - Status with Flags	1 octet	1	129
11 (0x0B)	0 (0x00)	Event	Binary Output Event - Any Variations		1	
11 (0x0B)	1 (0x01)	Event	Binary Output Event - Status	1 octet	1	129,130

Table 12-4—g3 double-bit binary input static objects

Group	Variation	Subset levels				Request (outstation must parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
3	0	x	x	x		—	—	▨	▨
3	0				✓	1 (READ)	00, 01, 06	▨	▨
3	0				✓	22 (ASSIGN_CLASS)	00, 01, 06	▨	▨
3	1	x	x	x		—	—	—	—
3	1				✓	1 (READ)	00, 01, 06	129 (RESPONSE)	00, 01
3	2	x	x	x		—	—	—	—
3	2				✓	1 (READ)	00, 01, 06	129 (RESPONSE)	00, 01

A.23.1.2.3 Notes

Read requests and responses shall use qualifier code 0x07
an outstation receives this request, it implicitly indicates
current time.

This object can be included in a write request. Write reque
value of 1 for this object. When an outstation receives th
wants to set the current time in the outstation.

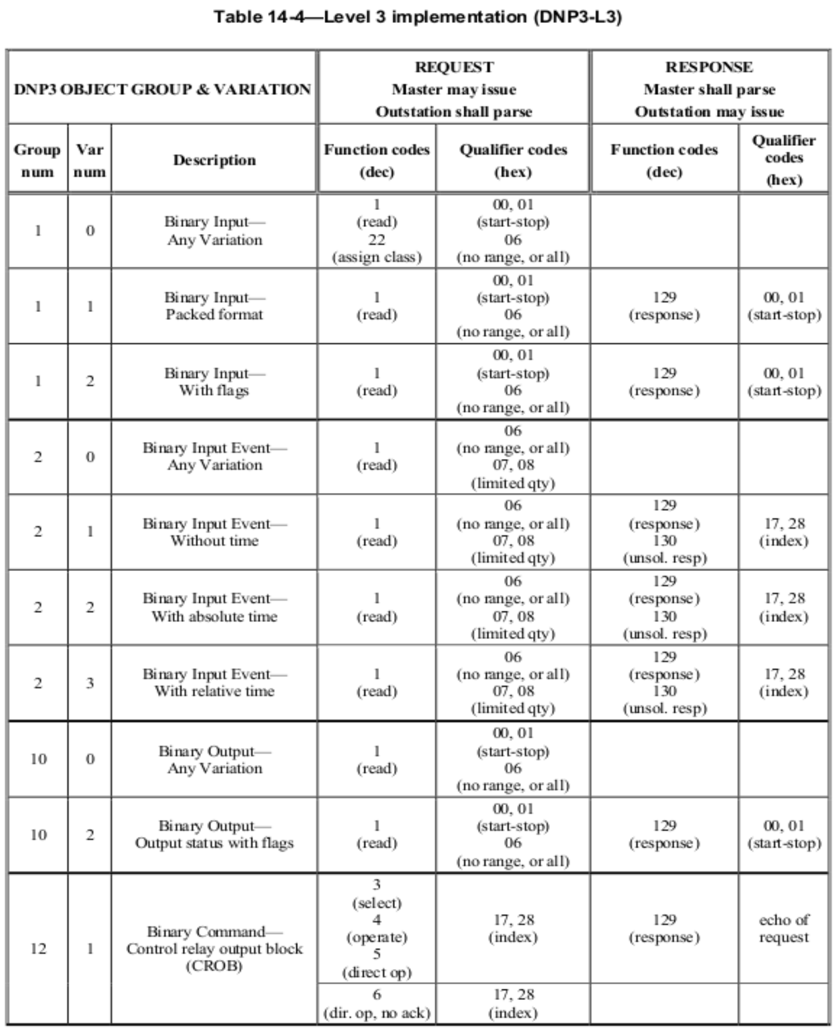


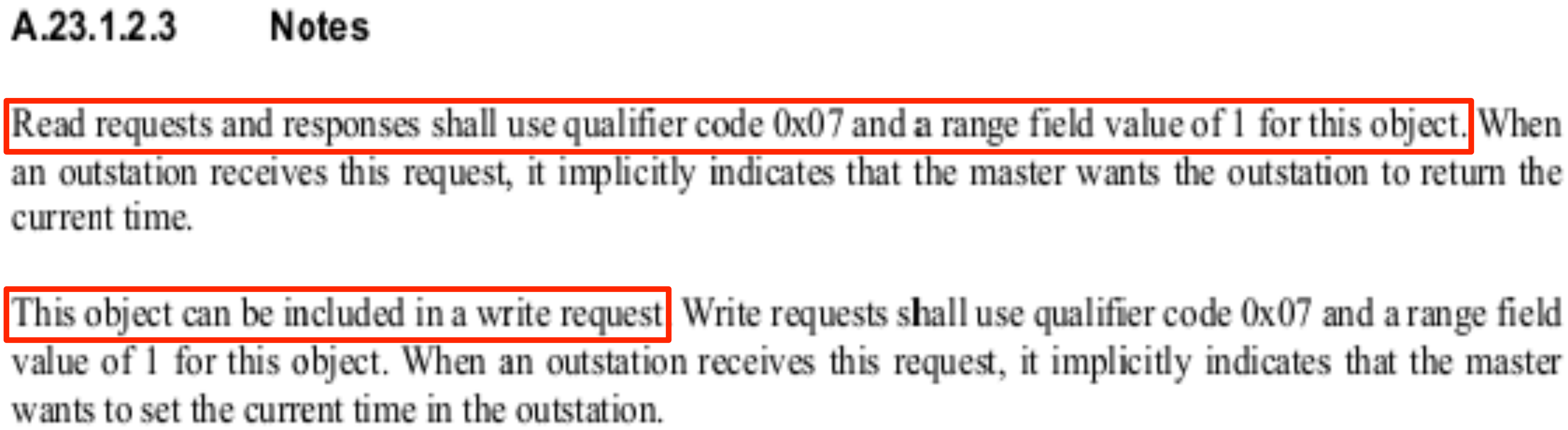
Table 14-4—Level 3 implementation (DNP3-L3)

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
1	0	Binary Input— Any Variation	1 (read) 22 (assign class)	00, 01 (start-stop) 06 (no range, or all)		
1	1	Binary Input— Packed format	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
1	2	Binary Input— With flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
2	0	Binary Input Event— Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
2	1	Binary Input Event— Without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	2	Binary Input Event— With absolute time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	3	Binary Input Event— With relative time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
10	0	Binary Output— Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
10	2	Binary Output— Output status with flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
12	1	Binary Command— Control relay output block (CROB)	3 (select) 4 (operate) 5 (direct op)	17, 28 (index)	129 (response)	echo of request
			6 (dir. op, no ack)	17, 28 (index)		

Syntax spills into semantics

```
// group 50 (times)...
g50v1_time_oblock = dnp3_p_single(G_V(TIME, TIME), time);
```

Object group 50: time and date



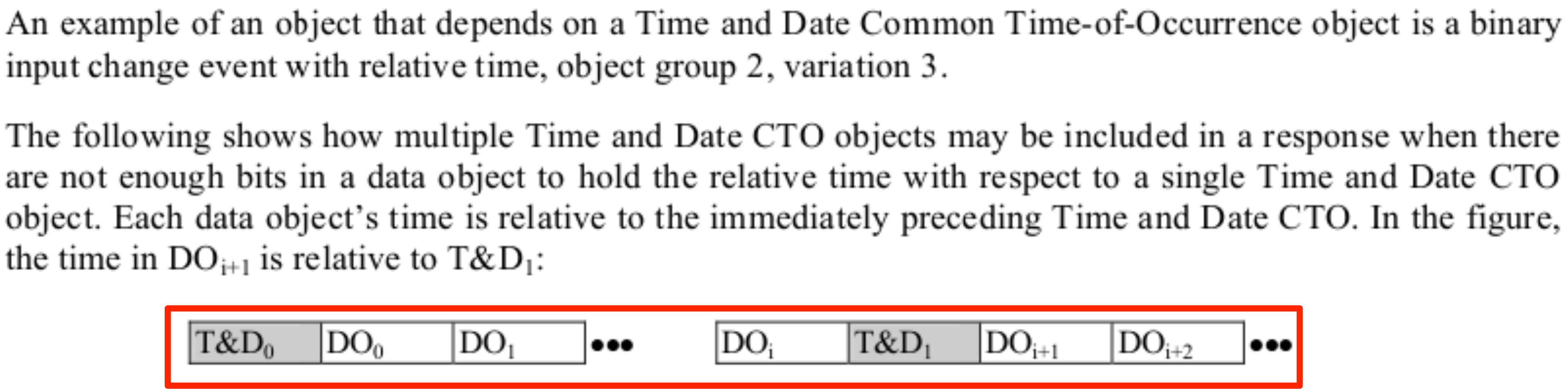
A.23.1.2.3 Notes

Read requests and responses shall use qualifier code 0x07 and a range field value of 1 for this object. When an outstation receives this request, it implicitly indicates that the master wants the outstation to return the current time.

This object can be included in a write request. Write requests shall use qualifier code 0x07 and a range field value of 1 for this object. When an outstation receives this request, it implicitly indicates that the master wants to set the current time in the outstation.

Syntax spills … where?

Object group 51: common time-of-occurance



An example of an object that depends on a Time and Date Common Time-of-Occurrence object is a binary input change event with relative time, object group 2, variation 3.

The following shows how multiple Time and Date CTO objects may be included in a response when there are not enough bits in a data object to hold the relative time with respect to a single Time and Date CTO object. Each data object’s time is relative to the immediately preceding Time and Date CTO. In the figure, the time in DO$_{i+1}$ is relative to T&D$_1$:

T&D$_0$	DO$_0$	DO$_1$	•••	DO$_i$	T&D$_1$	DO$_{i+1}$	DO$_{i+2}$	•••

“should the relative time variants generate an error unless preceded by a CTO object in the same message?”

Language Poison

- Range: (start,stop)
  - If we can't get this right...

- Better: (start,count), ala Modbus & IEC 104

- Would *ideally* like to avoid counts in the first place
  => Context-free!

Implementation Goals / Principles

- Be as grammatical as possible
  - Want to look like CFG, though we can't be

- Avoid code duplication (much abstraction)

- Capture DNP3's "true" syntax
  - Reject at syntax level what others may do later

Parser Combinators: look like grammars

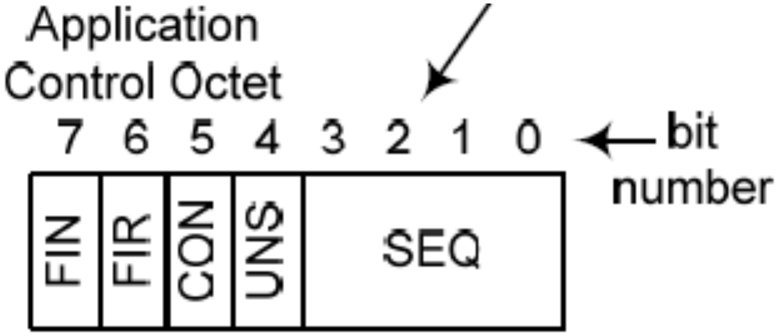
Have primitives

```
HParser *seqno = h_bits(4, false);
HParser *bit   = h_bits(1, false);
...
```

Combined to form higher-level structures

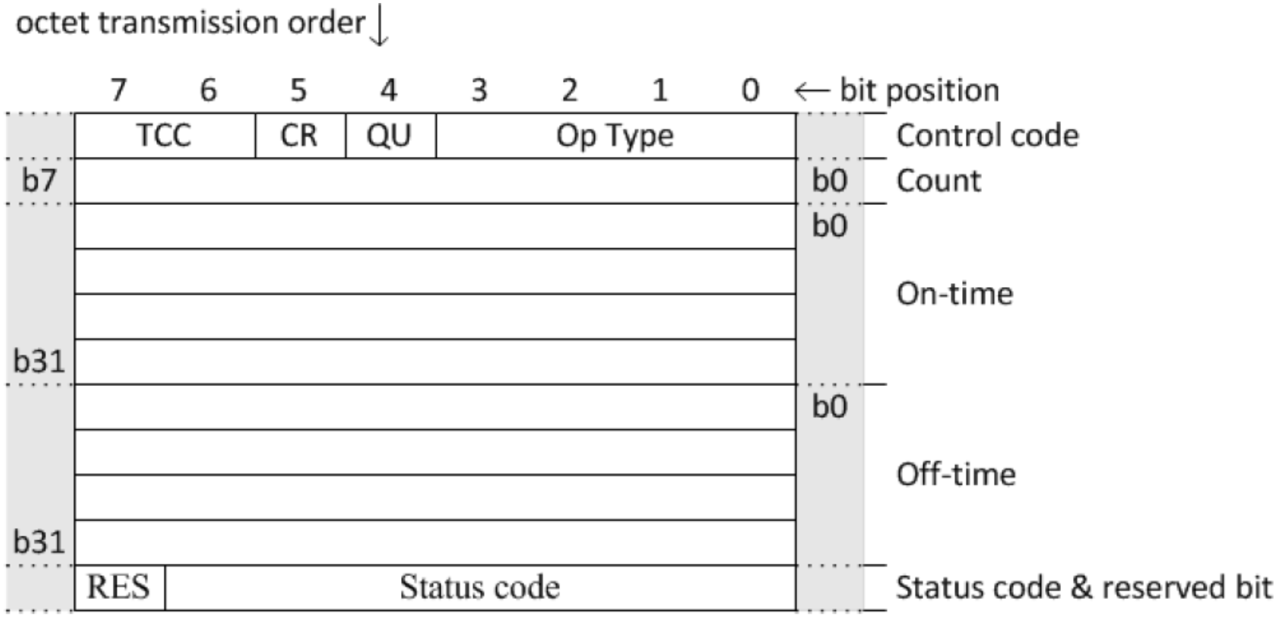
h_choice, h_many, h_many1, ...
define own combinators

Example – Fragment Header Flags



```
                /* --- uns,con,fin,fir --- */
conflags = h_sequence(bit,zro,one,one, NULL);    // CONFIRM
reqflags = h_sequence(zro,zro,one,one, NULL);    // always fin,fir!
unsflags = h_sequence(one,one,ign,ign, NULL);    // unsolicited
rspflags = h_sequence(zro,bit,bit,bit, NULL);
```

Example - CROB Object



```
crob = h_sequence(h_bits(4, false),  // op type
                      bit,           // queue flag
                      bit,           // clear flag
                      tcc,
                      h_uint8(),     // count
                      h_uint32(),    // on-time [ms]
                      h_uint32(),    // off-time [ms]
                      status,        // 7 bits
                      dnp3_p_reserved(1),
                      NULL));
```

Example – SELECT Function

```
pcb           = dnp3_p_g12v2_binoutcmd_pcb_oblock;
pcm           = dnp3_p_g12v3_binoutcmd_pcm_oblock;
select_pcb    = h_sequence(pcb, h_many1(pcm), NULL);
select_oblock = h_choice(select_pcb,
                         dnp3_p_g12v1_binoutcmd_crob_oblock,
                         dnp3_p_anaout_oblock,
                         NULL);
select         = h_many(select_oblock;
```

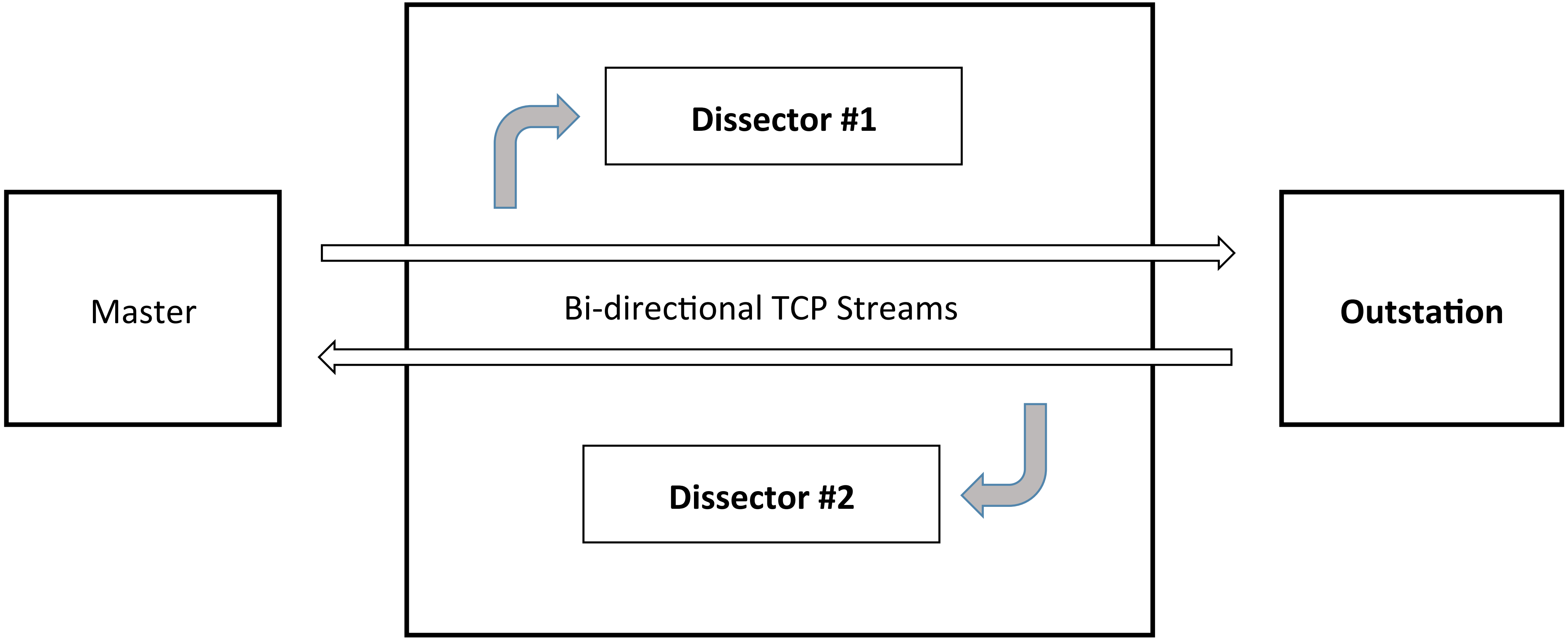
// empty select requests valid?

// is it valid to have many pcb-pcm blocks in the same request?

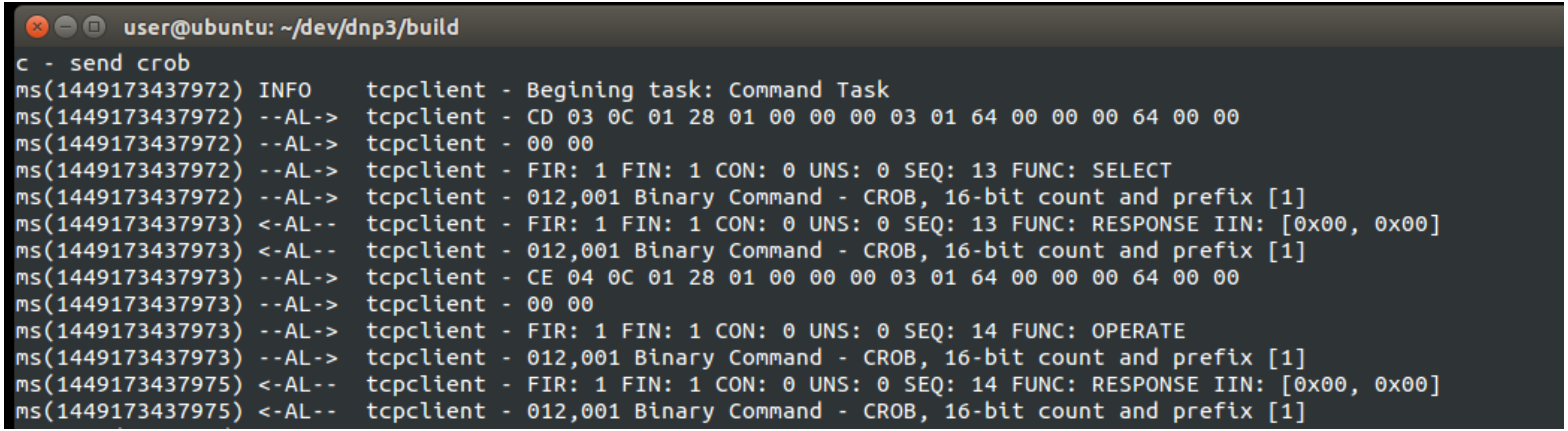
// ... to mix pcbs and crobs?

// langsec approach warns you of pitfalls!

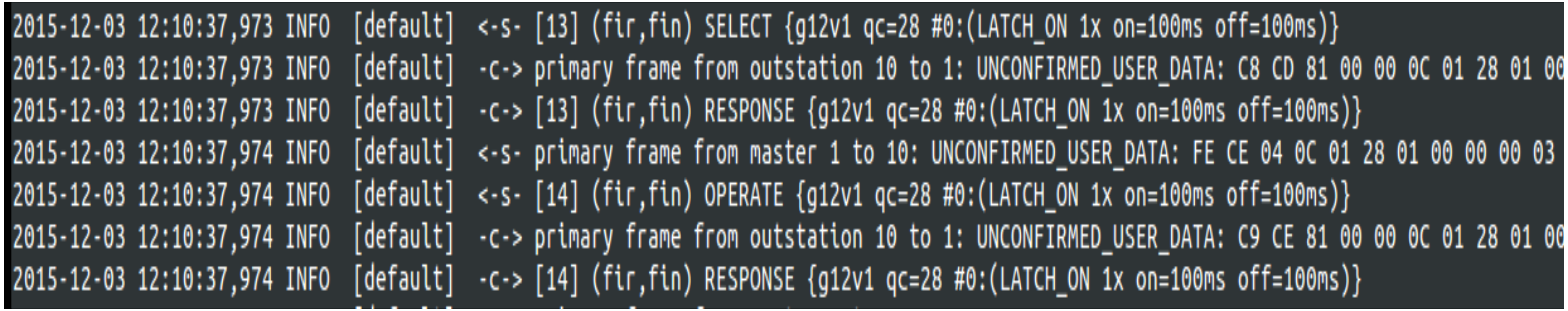
Practical application: Validating Proxy



Pretty printing of AST in log



```
c - send crob
ms(1449173437972) INFO    tcpclient - Begining task: Command Task
ms(1449173437972) --AL->  tcpclient - CD 03 0C 01 28 01 00 00 00 03 01 64 00 00 00 64 00 00
ms(1449173437972) --AL->  tcpclient - 00 00
ms(1449173437972) --AL->  tcpclient - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 13 FUNC: SELECT
ms(1449173437972) --AL->  tcpclient - 012,001 Binary Command - CROB, 16-bit count and prefix [1]
ms(1449173437973) <-AL--  tcpclient - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 13 FUNC: RESPONSE IIN: [0x00, 0x00]
ms(1449173437973) <-AL--  tcpclient - 012,001 Binary Command - CROB, 16-bit count and prefix [1]
ms(1449173437973) --AL->  tcpclient - CE 04 0C 01 28 01 00 00 00 03 01 64 00 00 00 64 00 00
ms(1449173437973) --AL->  tcpclient - 00 00
ms(1449173437973) --AL->  tcpclient - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 14 FUNC: OPERATE
ms(1449173437973) --AL->  tcpclient - 012,001 Binary Command - CROB, 16-bit count and prefix [1]
ms(1449173437975) <-AL--  tcpclient - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 14 FUNC: RESPONSE IIN: [0x00, 0x00]
ms(1449173437975) <-AL--  tcpclient - 012,001 Binary Command - CROB, 16-bit count and prefix [1]
```



```
2015-12-03 12:10:37,973 INFO  [default]  <-s- [13] (fir,fin) SELECT {g12v1 qc=28 #0:(LATCH_ON 1x on=100ms off=100ms)}
2015-12-03 12:10:37,973 INFO  [default]  -c-> primary frame from outstation 10 to 1: UNCONFIRMED_USER_DATA: C8 CD 81 00 00 0C 01 28 01 00
2015-12-03 12:10:37,973 INFO  [default]  -c-> [13] (fir,fin) RESPONSE {g12v1 qc=28 #0:(LATCH_ON 1x on=100ms off=100ms)}
2015-12-03 12:10:37,974 INFO  [default]  <-s- primary frame from master 1 to 10: UNCONFIRMED_USER_DATA: FE CE 04 0C 01 28 01 00 00 00 03
2015-12-03 12:10:37,974 INFO  [default]  <-s- [14] (fir,fin) OPERATE {g12v1 qc=28 #0:(LATCH_ON 1x on=100ms off=100ms)}
2015-12-03 12:10:37,974 INFO  [default]  -c-> primary frame from outstation 10 to 1: UNCONFIRMED_USER_DATA: C9 CE 81 00 00 0C 01 28 01 00
2015-12-03 12:10:37,974 INFO  [default]  -c-> [14] (fir,fin) RESPONSE {g12v1 qc=28 #0:(LATCH_ON 1x on=100ms off=100ms)}
```

Validation: familiar tools/techniques

- Unit tests, Unit tests, Unit tests
- Tests based on common DNP3 implementation mistakes
- Dynamic analysis with Valgrind
- Fuzzing: coverage-guided (AFL) and model-based (Aegis)
- No static analysis, but multiple compilers including Clang

No silver bullet, but correct tactic

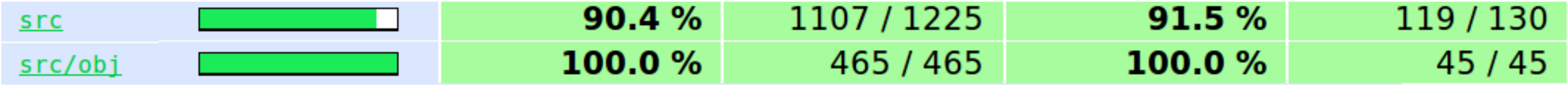
- Langsec approach doesn’t guarantee success, but provides a **disciplined roadmap** for success

- Traditional testing techniques are just as important, but Langsec gives them more order (when to test what? What to test for? Factor your code so that it’s testable—parser before processing)

- Well-factored parsers will be more maintainable and extensible

Write tests as you write production code.

```
// mixing CROBs, analog output, and PCBs

check_parse(dnp3_p_app_request,
                                                  "\xC3\x03\x0C
\x02\x07\x01\x41\x03\xF4\x01\x00\x00\xD0\x07\x00\x00\x0                    "\x0C
\x03\x00\x05\x0F\x21\x04"
          "\x29\x01\x17\x01\x01\x12\x34\x56\x78\x00", 34,

           "[3] (fir,fin) SELECT {g12v2 qc=07 (CLOSE PULSE_ON 3x on=500ms off=2000ms)}"
           " {g12v3 qc=00 #5..15: 1 0 0 0 0 1 0 0 0 0 1}"
           " {g41v1 qc=17 #1:2018915346}");
```

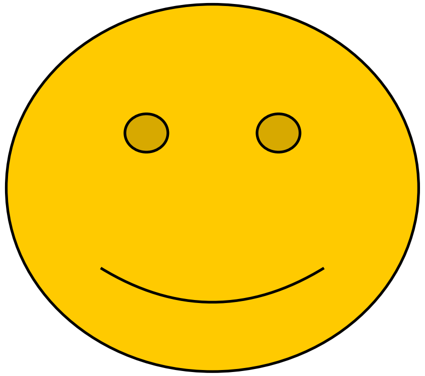


src	🟩	**90.4 %**	1107 / 1225	**91.5 %**	119 / 130
src/obj	🟩	**100.0 %**	465 / 465	**100.0 %**	45 / 45

Unit tests for known poison

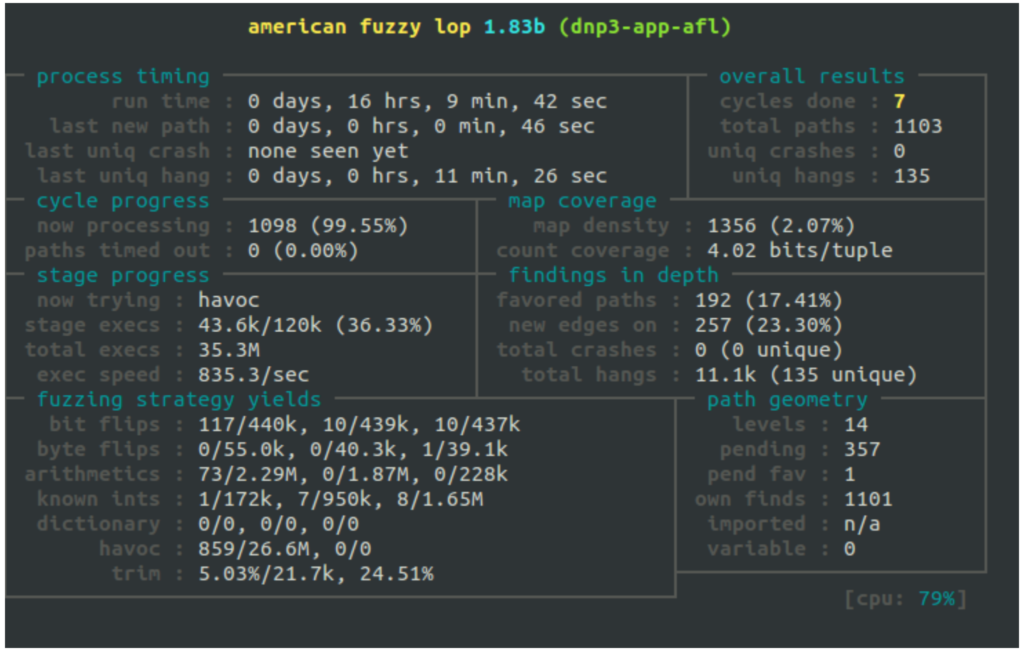
```
// 4-byte max range - start = 0, stop = 0xFFFFFFFF
check_parse(dnp3_p_app_response,
            "\x00\x81\x00\x00\x1E\x02\x02\x00\x00\x00\x00\xFF\xFF\xFF\xFF", 15,
            "PARAM_ERROR on [0] RESPONSE");
```

```
static HParsedToken *act_range(const HParseResult *p, void *user)
{
   // p->ast = (start, stop)
   uint64_t start = H_FIELD_UINT(0);
   uint64_t stop  = H_FIELD_UINT(1);

   assert(start <= stop);
   assert(stop - start < SIZE_MAX);
   return H_MAKE_UINT(stop - start + 1);
}
```

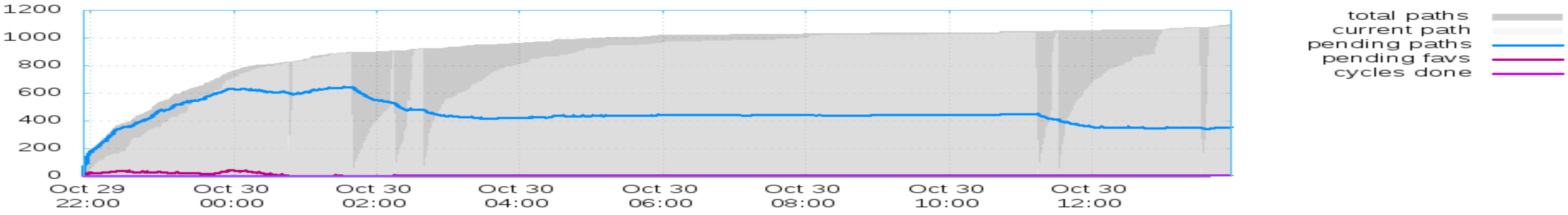


American fuzzy lop (AFL)

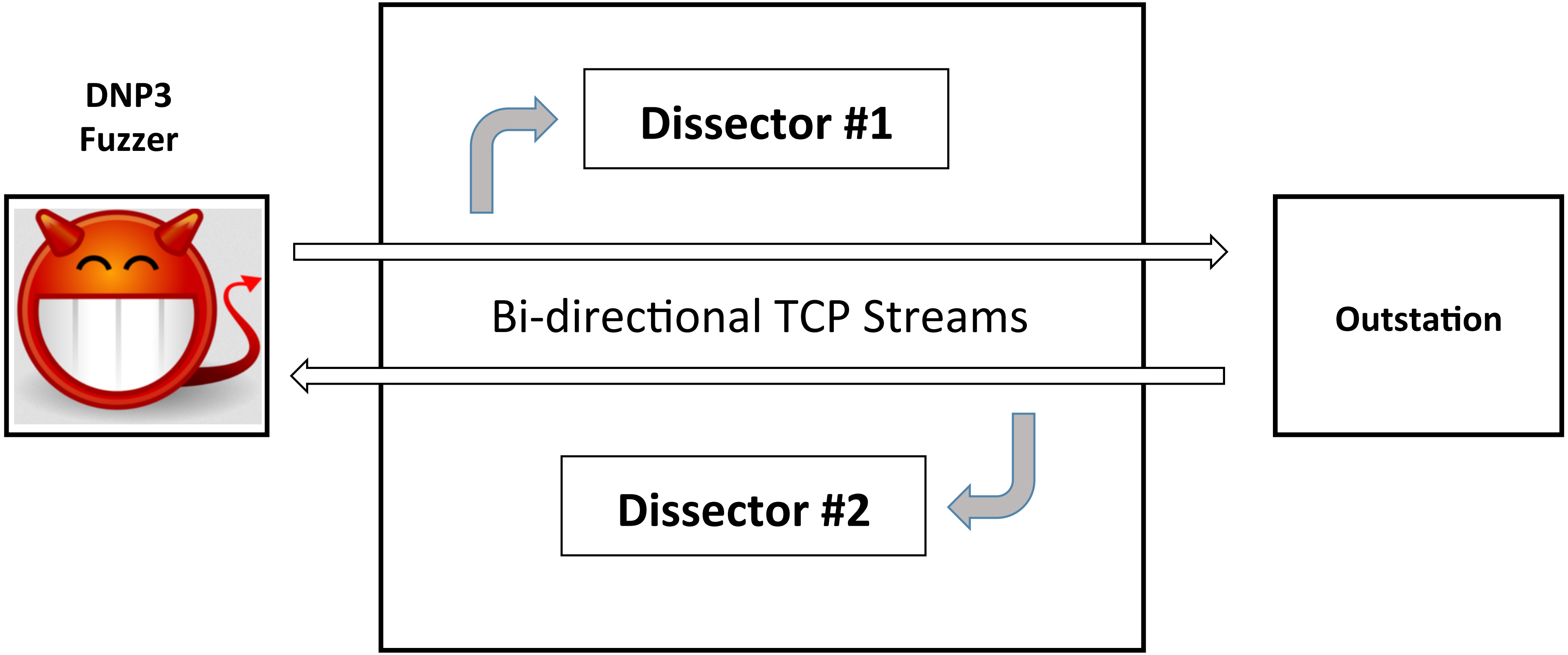
- Generic coverage-guided fuzzing
- Program must accept input from stdin



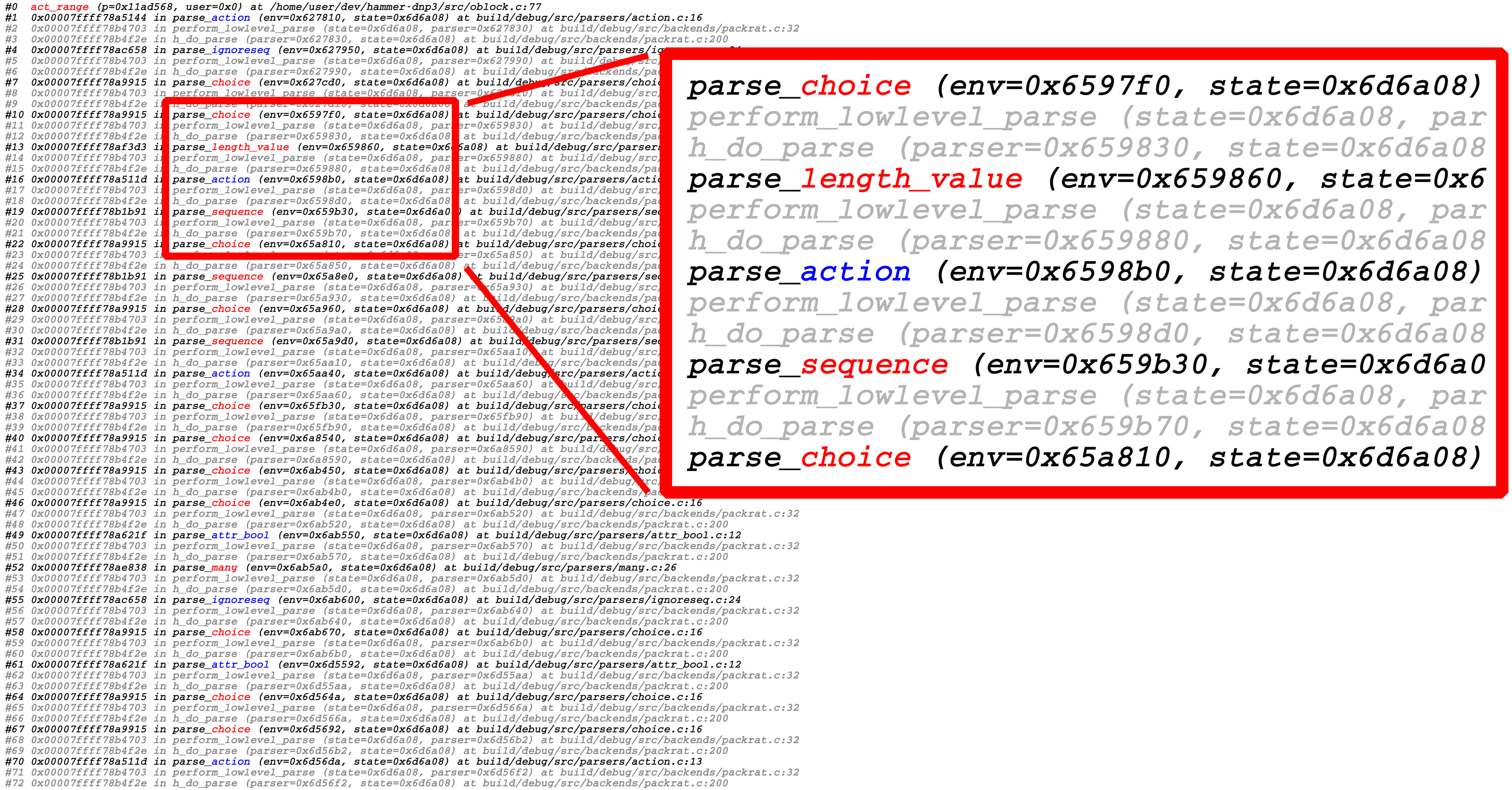
```
                american fuzzy lop 1.83b (dnp3-app-afl)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 16 hrs, 9 min, 42 sec      │  cycles done : 7      │
│   last new path : 0 days, 0 hrs, 0 min, 46 sec       │  total paths : 1103   │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : 0 days, 0 hrs, 11 min, 26 sec      │   uniq hangs : 135    │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 1098 (99.55%)     │    map density : 1356 (2.07%)          │
│ paths timed out : 0 (0.00%)         │ count coverage : 4.02 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : havoc                 │ favored paths : 192 (17.41%)           │
│ stage execs : 43.6k/120k (36.33%)   │  new edges on : 257 (23.30%)           │
│ total execs : 35.3M                 │ total crashes : 0 (0 unique)           │
│  exec speed : 835.3/sec             │   total hangs : 11.1k (135 unique)     │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 117/440k, 10/439k, 10/437k            │    levels : 14         │
│  byte flips : 0/55.0k, 0/40.3k, 1/39.1k             │   pending : 357        │
│ arithmetics : 73/2.29M, 0/1.87M, 0/228k             │  pend fav : 1          │
│  known ints : 1/172k, 7/950k, 8/1.65M               │ own finds : 1101       │
│  dictionary : 0/0, 0/0, 0/0                         │  imported : n/a        │
│       havoc : 859/26.6M, 0/0                        │  variable : 0          │
│        trim : 5.03%/21.7k, 24.51%                   ├────────────────────────┘
└─────────────────────────────────────────────────────┘             [cpu: 79%]
```



Fuzzing in observe-only mode



Challenges - Deep, generic stack traces



```
#0  act_range (p=0x11ad568, user=0x0) at /home/user/dev/hammer-dnp3/src/oblock.c:77
#1  0x00007ffff78a5144 in parse_action (env=0x627810, state=0x6d6a08) at build/debug/src/parsers/action.c:16
#2  0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x627830) at build/debug/src/backends/packrat.c:32
#3  0x00007ffff78b4f2e in h_do_parse (parser=0x627830, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
#4  0x00007ffff78ac658 in parse_ignoreseq (env=0x627950, state=0x6d6a08) at build/debug/src/parsers/ig
#5  0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x627990) at build/deb   src/
#6  0x00007ffff78b4f2e in h_do_parse (parser=0x627990, state=0x6d6a08) at build/debug/s  ackends/pa
#7  0x00007ffff78a9915 in parse_choice (env=0x627cd0, state=0x6d6a08) at build/deb  src/parsers/choi
#8  0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6    10) at build/debug/src/
#9  0x00007ffff78b4f2e i                                                         build/debug/src/backends/pa
#10 0x00007ffff78a9915 i  parse_choice (env=0x6597f0, state=0x6d6a08) at build/debug/src/parsers/choi
#11 0x00007ffff78b4703 i  perform_lowlevel_parse (state=0x6d6a08, par er=0x659830) at build/debug/src/
#12 0x00007ffff78b4f2e i  h_do_parse (parser=0x659830, state=0x6d6a08  at build/debug/src/backends/pa
#13 0x00007ffff78af3d3 i  parse_length_value (env=0x659860, state=0x6 6a08) at build/debug/src/parsers
#14 0x00007ffff78b4703 i  perform_lowlevel_parse (state=0x6d6a08, par er=0x659880) at build/debug/src/
#15 0x00007ffff78b4f2e i  h_do_parse (parser=0x659880, state=0x6d6a08  at build/debug/src/backends/pa
#16 0x00007ffff78a511d i  parse_action (env=0x6598b0, state=0x6d6a08) at build/debug/src/parsers/acti
#17 0x00007ffff78b4703 i  perform_lowlevel_parse (state=0x6d6a08, par er=0x6598d0) at build/debug/src/
#18 0x00007ffff78b4f2e i  h_do_parse (parser=0x6598d0, state=0x6d6a08  at build/debug/src/backends/pa
#19 0x00007ffff78b1b91 i  parse_sequence (env=0x659b30, state=0x6d6a0 ) at build/debug/src/parsers/se
#20 0x00007ffff78b4703 i  perform_lowlevel_parse (state=0x6d6a08, par er=0x659b70) at build/debug/src/
#21 0x00007ffff78b4f2e i  h_do_parse (parser=0x659b70, state=0x6d6a08  at build/debug/src/backends/pa
#22 0x00007ffff78a9915 i  parse_choice (env=0x65a810, state=0x6d6a08) at build/debug/src/parsers/choi
#23 0x00007ffff78b4703 i                                            er=0x65a850) at build/debug/src/
#24 0x00007ffff78b4f2e in h_do_parse (parser=0x65a850, state=0x6d6a08)  at build/debug/src/backends/pa
#25 0x00007ffff78b1b91 in parse_sequence (env=0x65a8e0, state=0x6d6a08)  t build/debug/src/parsers/se
#26 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser= x65a930) at build/debug/src/
#27 0x00007ffff78b4f2e in h_do_parse (parser=0x65a930, state=0x6d6a08) at  ild/debug/src/backends/pa
#28 0x00007ffff78a9915 in parse_choice (env=0x65a960, state=0x6d6a08) at bui d/debug/src/parsers/choi
#29 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x65 a0) at build/debug/src/
#30 0x00007ffff78b4f2e in h_do_parse (parser=0x65a9a0, state=0x6d6a08) at buil debug/src/backends/pa
#31 0x00007ffff78b1b91 in parse_sequence (env=0x65a9d0, state=0x6d6a08) at buil /debug/src/parsers/se
#32 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x65aa10  at build/debug/src/
#33 0x00007ffff78b4f2e in h_do_parse (parser=0x65aa10, state=0x6d6a08) at build/de  g/src/backends/pa
#34 0x00007ffff78a511d in parse_action (env=0x65aa40, state=0x6d6a08) at build/debug src/parsers/acti
#35 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x65aa60) at  uild/debug/src/
#36 0x00007ffff78b4f2e in h_do_parse (parser=0x65aa60, state=0x6d6a08) at build/debug/s c/backends/pa
#37 0x00007ffff78a9915 in parse_choice (env=0x65fb30, state=0x6d6a08) at build/debug/src parsers/choi
#38 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x65fb90) at bui /debug/src/
#39 0x00007ffff78b4f2e in h_do_parse (parser=0x65fb90, state=0x6d6a08) at build/debug/src/b ckends/pa
#40 0x00007ffff78a9915 in parse_choice (env=0x6a8540, state=0x6d6a08) at build/debug/src/par ers/choi
#41 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6a8590) at build/de ug/src/
#42 0x00007ffff78b4f2e in h_do_parse (parser=0x6a8590, state=0x6d6a08) at build/debug/src/backe ds/pa
#43 0x00007ffff78a9915 in parse_choice (env=0x6ab450, state=0x6d6a08) at build/debug/src/parsers choi
#44 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6ab4b0) at build/debug  rc/
#45 0x00007ffff78b4f2e in h_do_parse (parser=0x6ab4b0, state=0x6d6a08) at build/debug/src/backends  a
#46 0x00007ffff78a9915 in parse_choice (env=0x6ab4e0, state=0x6d6a08) at build/debug/src/parsers/choice.c:16
#47 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6ab520) at build/debug/src/backends/packrat.c:32
#48 0x00007ffff78b4f2e in h_do_parse (parser=0x6ab520, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
#49 0x00007ffff78a621f in parse_attr_bool (env=0x6ab550, state=0x6d6a08) at build/debug/src/parsers/attr_bool.c:12
#50 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6ab570) at build/debug/src/backends/packrat.c:32
#51 0x00007ffff78b4f2e in h_do_parse (parser=0x6ab570, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
#52 0x00007ffff78ae838 in parse_many (env=0x6ab5a0, state=0x6d6a08) at build/debug/src/parsers/many.c:26
#53 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6ab5d0) at build/debug/src/backends/packrat.c:32
#54 0x00007ffff78b4f2e in h_do_parse (parser=0x6ab5d0, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
#55 0x00007ffff78ac658 in parse_ignoreseq (env=0x6ab600, state=0x6d6a08) at build/debug/src/parsers/ignoreseq.c:24
#56 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6ab640) at build/debug/src/backends/packrat.c:32
#57 0x00007ffff78b4f2e in h_do_parse (parser=0x6ab640, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
#58 0x00007ffff78a9915 in parse_choice (env=0x6ab670, state=0x6d6a08) at build/debug/src/parsers/choice.c:16
#59 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6ab6b0) at build/debug/src/backends/packrat.c:32
#60 0x00007ffff78b4f2e in h_do_parse (parser=0x6ab6b0, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
#61 0x00007ffff78a621f in parse_attr_bool (env=0x6d5592, state=0x6d6a08) at build/debug/src/parsers/attr_bool.c:12
#62 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6d55aa) at build/debug/src/backends/packrat.c:32
#63 0x00007ffff78b4f2e in h_do_parse (parser=0x6d55aa, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
#64 0x00007ffff78a9915 in parse_choice (env=0x6d564a, state=0x6d6a08) at build/debug/src/parsers/choice.c:16
#65 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6d566a) at build/debug/src/backends/packrat.c:32
#66 0x00007ffff78b4f2e in h_do_parse (parser=0x6d566a, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
#67 0x00007ffff78a9915 in parse_choice (env=0x6d5692, state=0x6d6a08) at build/debug/src/parsers/choice.c:16
#68 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6d56b2) at build/debug/src/backends/packrat.c:32
#69 0x00007ffff78b4f2e in h_do_parse (parser=0x6d56b2, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
#70 0x00007ffff78a511d in parse_action (env=0x6d56da, state=0x6d6a08) at build/debug/src/parsers/action.c:13
#71 0x00007ffff78b4703 in perform_lowlevel_parse (state=0x6d6a08, parser=0x6d56f2) at build/debug/src/backends/packrat.c:32
#72 0x00007ffff78b4f2e in h_do_parse (parser=0x6d56f2, state=0x6d6a08) at build/debug/src/backends/packrat.c:200
```

```
parse_choice (env=0x6597f0, state=0x6d6a08)
perform_lowlevel_parse (state=0x6d6a08, par
h_do_parse (parser=0x659830, state=0x6d6a08
parse_length_value (env=0x659860, state=0x6
perform_lowlevel_parse (state=0x6d6a08, par
h_do_parse (parser=0x659880, state=0x6d6a08
parse_action (env=0x6598b0, state=0x6d6a08)
perform_lowlevel_parse (state=0x6d6a08, par
h_do_parse (parser=0x6598d0, state=0x6d6a08
parse_sequence (env=0x659b30, state=0x6d6a0
perform_lowlevel_parse (state=0x6d6a08, par
h_do_parse (parser=0x659b70, state=0x6d6a08
parse_choice (env=0x65a810, state=0x6d6a08)
```

Some Lessons Learned

- DNP3 is obviously well-intentioned :)
  - Wants syntax to be simple

- Unfortunately ends up doing it wrong :'(
  - "Uniform" syntax not so uniform

- Could almost be context-free

- Start/stop based index syntax is just plain dangerous.

Discoveries

- Several design/clarification questions
  - correct to ignore FCB on secondary frames?
  - is there a minimum number of bytes in the transport payload?
  - ….
- Spec bugs/issues
  - AN2013-004b: RESPONSE can also include g120v1
  - should status bits be 8 on anaout, but 7 everywhere else?“
  - ….

Future work

- Language subsetting, i.e. constraining grammar via configuration
- Structs -> output (aka un-parsing)
- Open questions WRT to protocol particularities
- Missing features in parser
  - g120 – authentication structures
  - g70 - File transfer
- Proxy that processes multiple sessions

OS protections for well-separated parsers

- Parser is the most dangerous part of the program
  - Most memory corruptions and exploits occur here
- When properly separated, it can be isolated by OS means
- **ELFbac**: a Linux kernel-based memory isolation for code and data in ELF binary files sections
  - Enforces ACLs between code and data units
    - E.g.: only the parser reads raw input buffers
  - Compatible with Grsecurity/PaX patches
  - Exists for x86 and ARM (public release this January)
- Works for our DNP3 proxy!

Fin - Questions?

https://github.com/pesco/dnp3
https://github.com/sergeybratus/proxy
open source, BSD license

3rd LangSec IEEE S&P workshop: http://spw16.langsec.org/

TUHH
Technische Universität Hamburg-Harburg

DARTMOUTH
VOX CLAMANTIS IN DESERTO
1769

AUTOMATAK

SEL
SCHWEITZER ENGINEERING LABORATORIES, INC.