Replacing Flow Dissector with PANDA Parser

Tom Herbert - SiPanda
Pedro Tammela - Mojatatu Networks

Netdev 0x15

Agenda

- Flow dissector
- The PANDA parser
- PANDA parser in the kernel and the PANDA classifier
- Performance results (flow dissector vs. PANDA)
- Futures
- PANDA project information

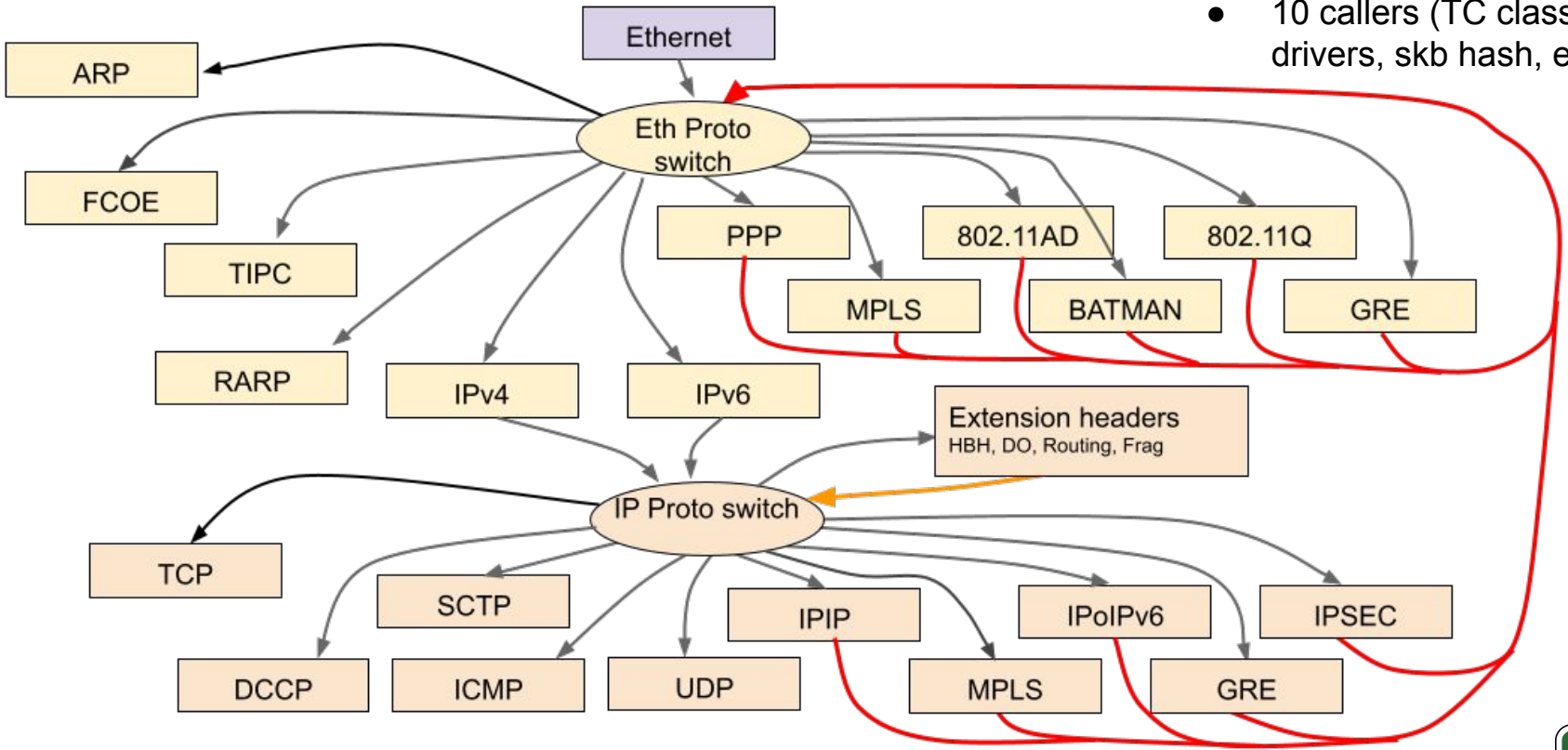
Flow dissector background

Flow dissector (**flō dis·sec'tor**) is a routine that parses metadata out of the packets. It's used in various places in the kernel networking subsystem (RFS, flow hash, etc.).

- **rps: Receive Packet Steering** (Tom Herbert, March 2010)
  - get_rps_cpu: Create tuple hash from IPv4/IPv6 and transport ports (UDP, TCP, etc.)
- **net: introduce skb_flow_dissect()** (Eric Dumazet, November, 2011)
  - net: Introduce flow_keys to store metadata
  - Parse IPv[46], 802.1Q, PPP, GRE, IPIP, transport headers for port offsets
- **flow_dissector: introduce programmable flow_dissector** (Jiri Pirko, May 2017)
  - Control what keys, i.e. metadata, is extracted
- **flow_dissector: implements flow dissector BPF hook** (Peter Penkov, September, 2018)
  - If BPF is attached to flow dissector, that is run in lieu of flow dissector code (i.e. not integrated)

Flow dissector today

- net/core/flow_dissector.c
- 1200 LOC
- BPF, tunnel support*
- 10 callers (TC classifiers, drivers, skb hash, etc.)

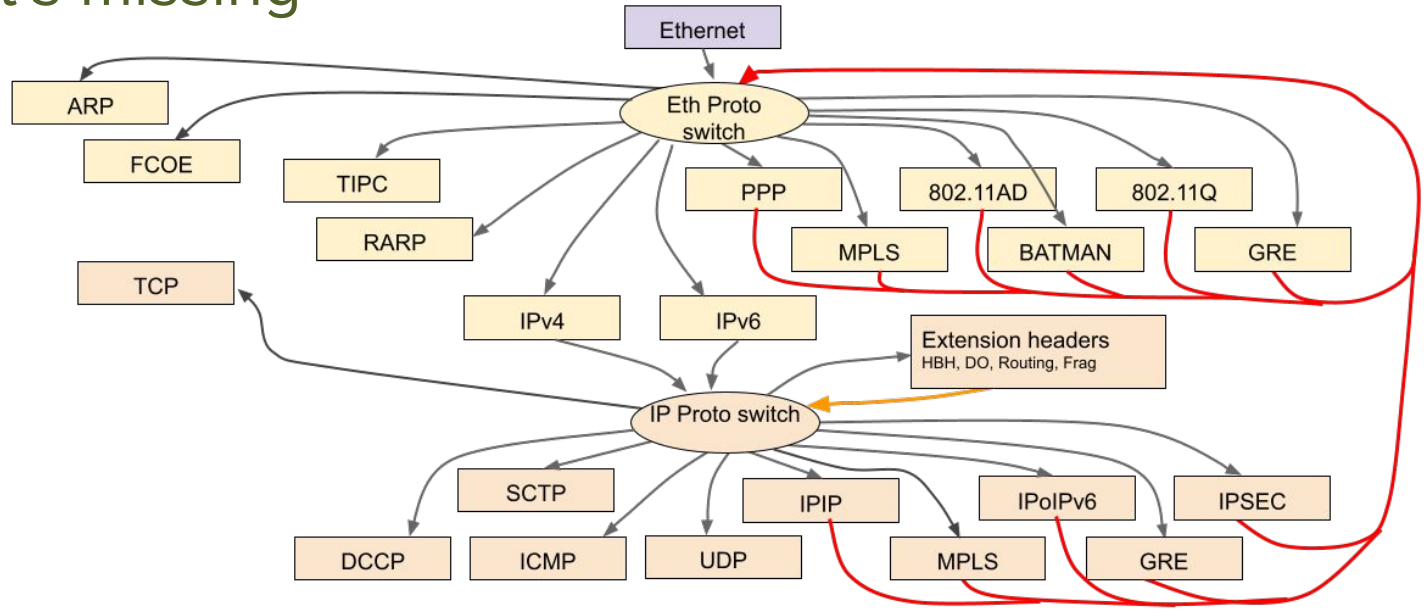


Main function: skb_flow_dissect

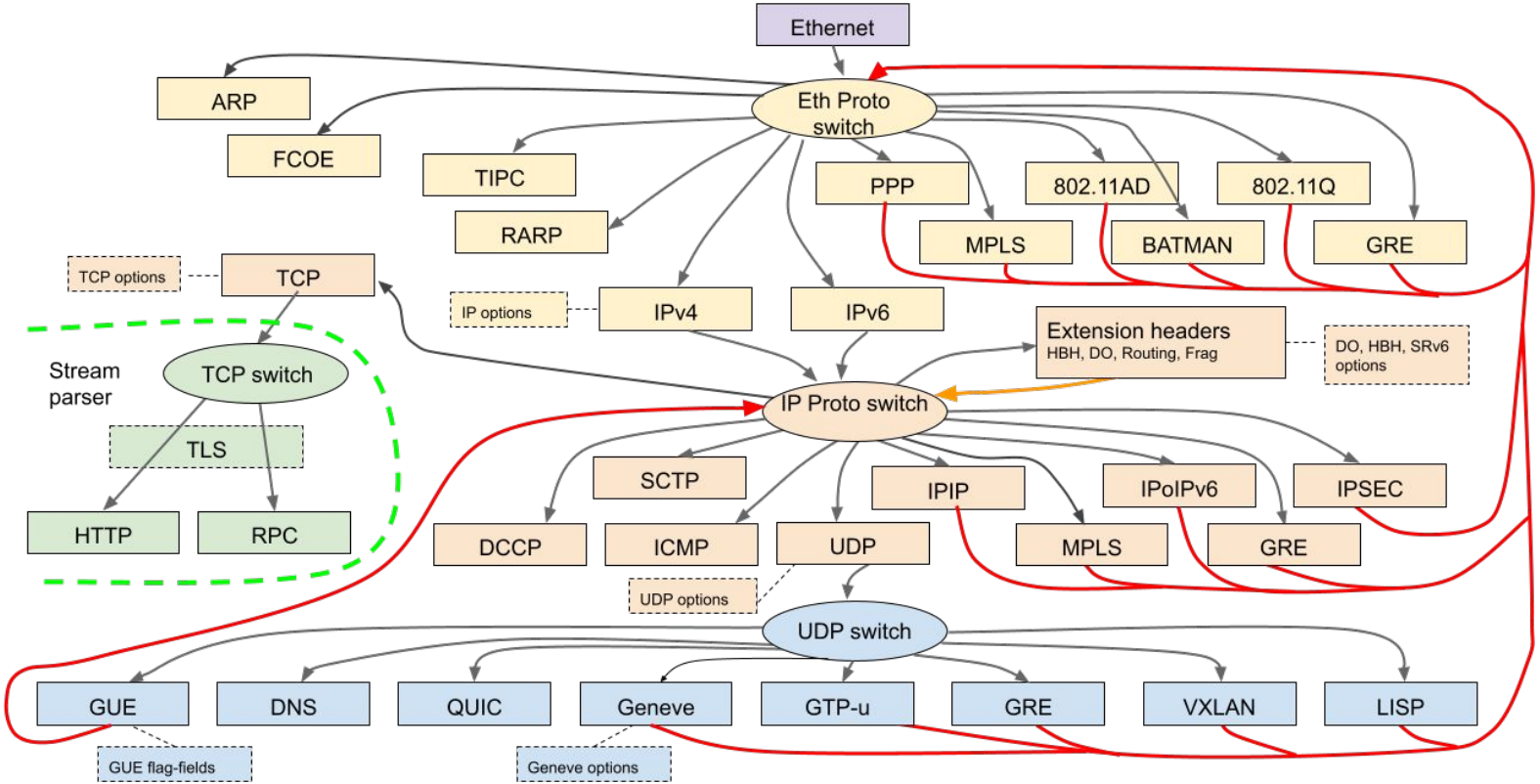
The function you love to hate!

- Frankly, it's spaghetti code!
- Bookkeeping has been prone to errors
- Doesn't parse UDP payloads or encaps
- Adding new protocols is a pain
  - Extends to userspace, e.g. tc-flower
  - Extends to programmable HW since API is constrained
- Rigid metadata structure means loss of information
  - e.g. what if someone wanted to know about a middle layer encap?

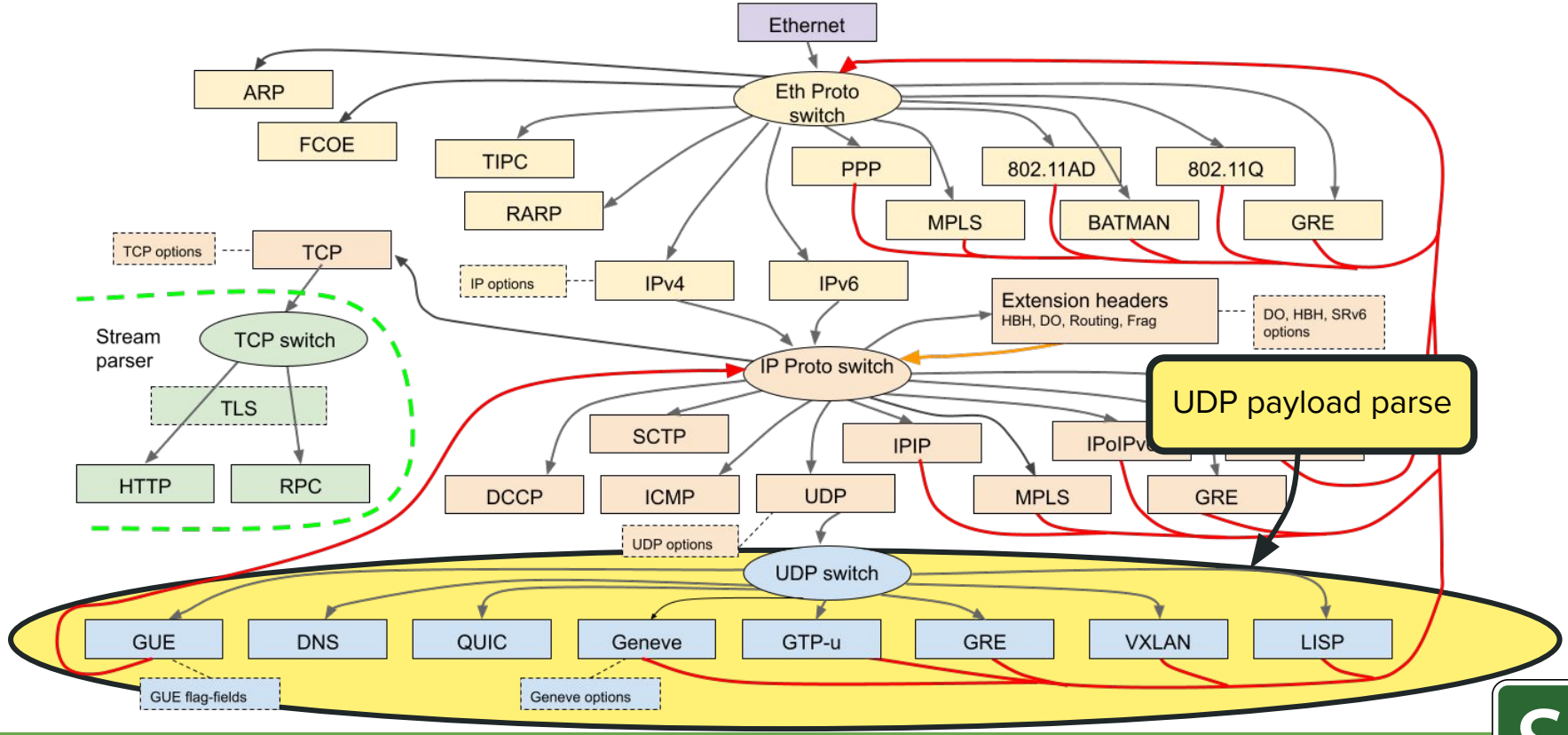
So what's missing



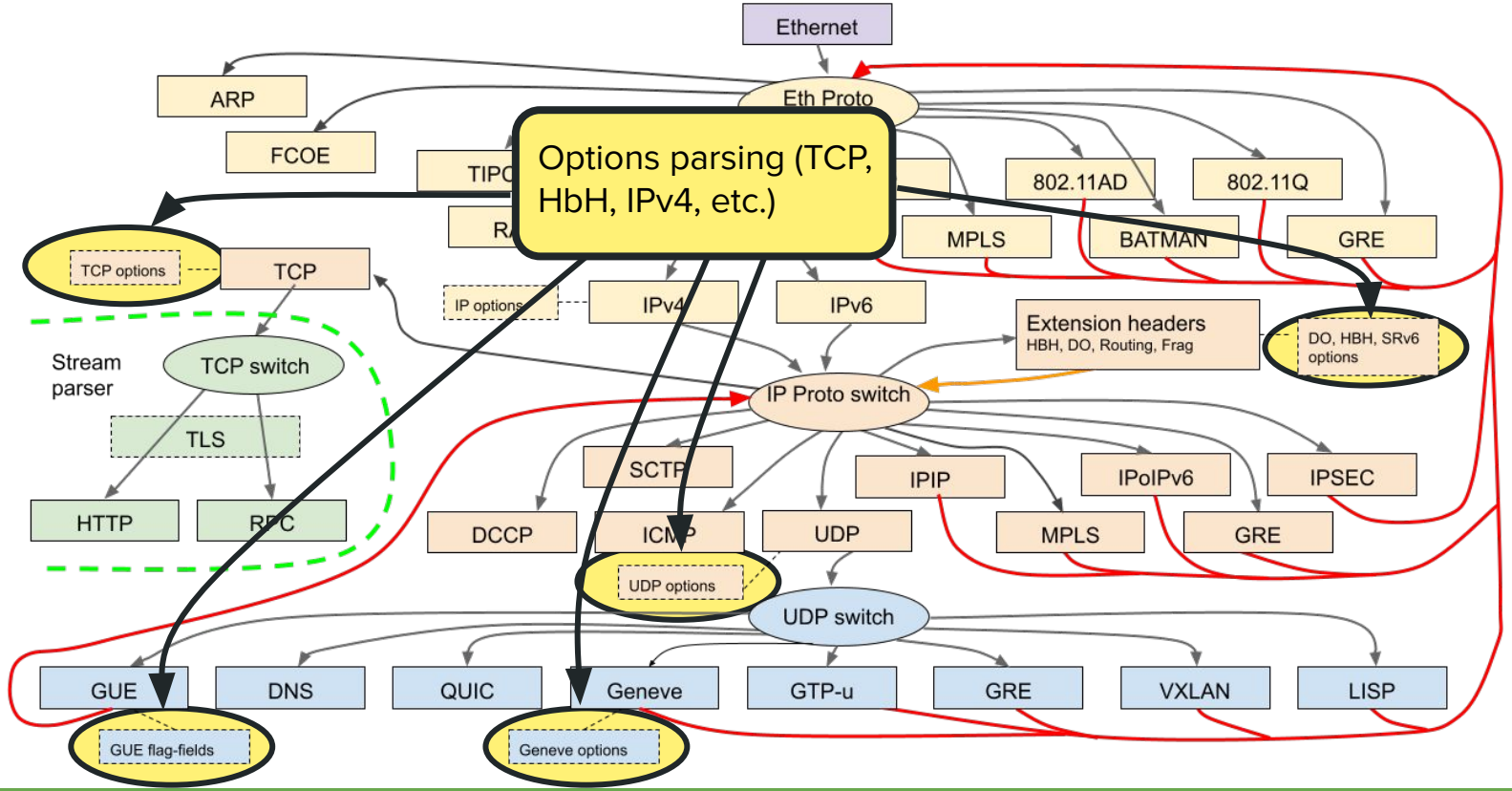
So what's missing



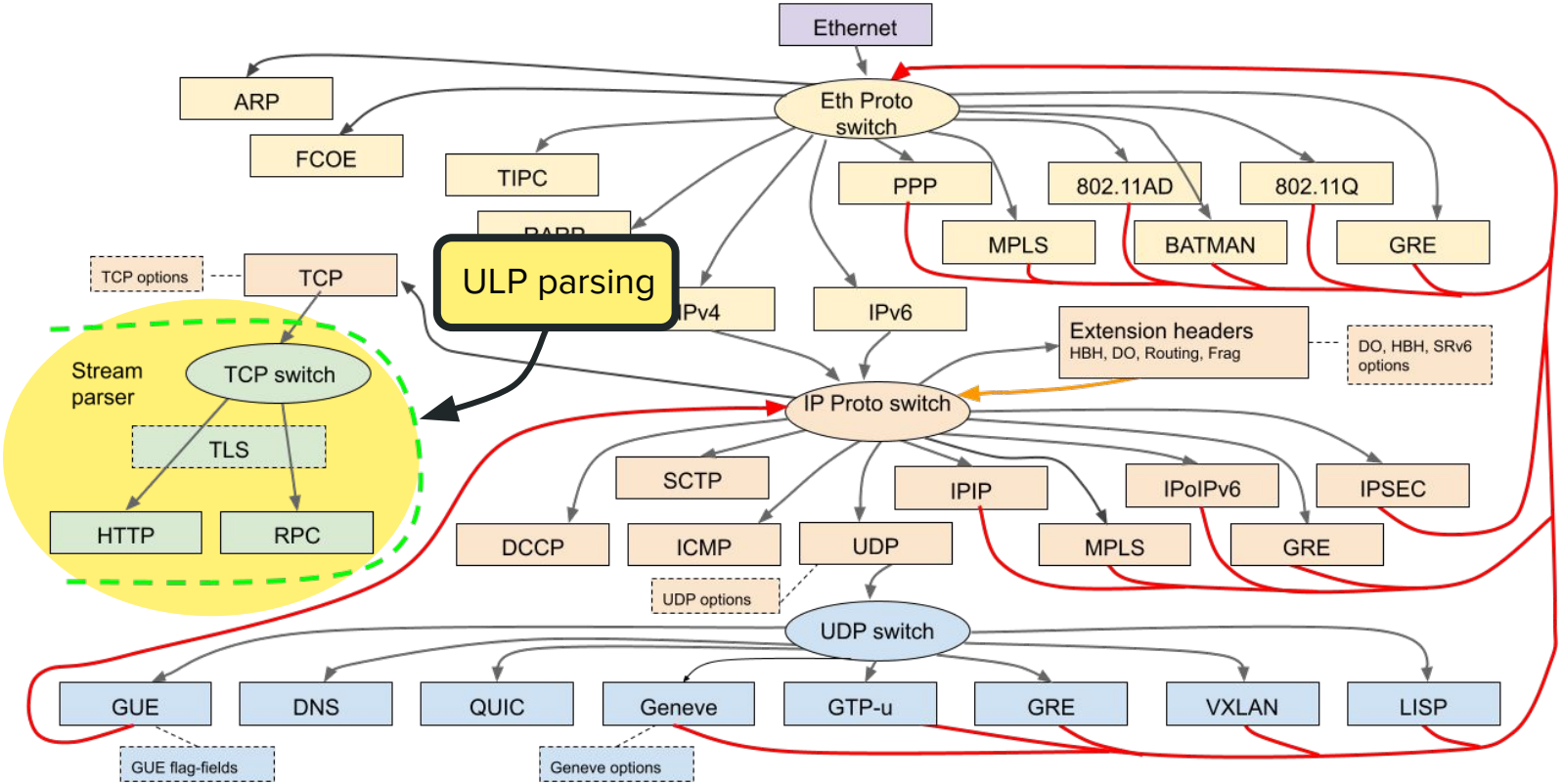
So what's missing



So what's missing



So what's missing



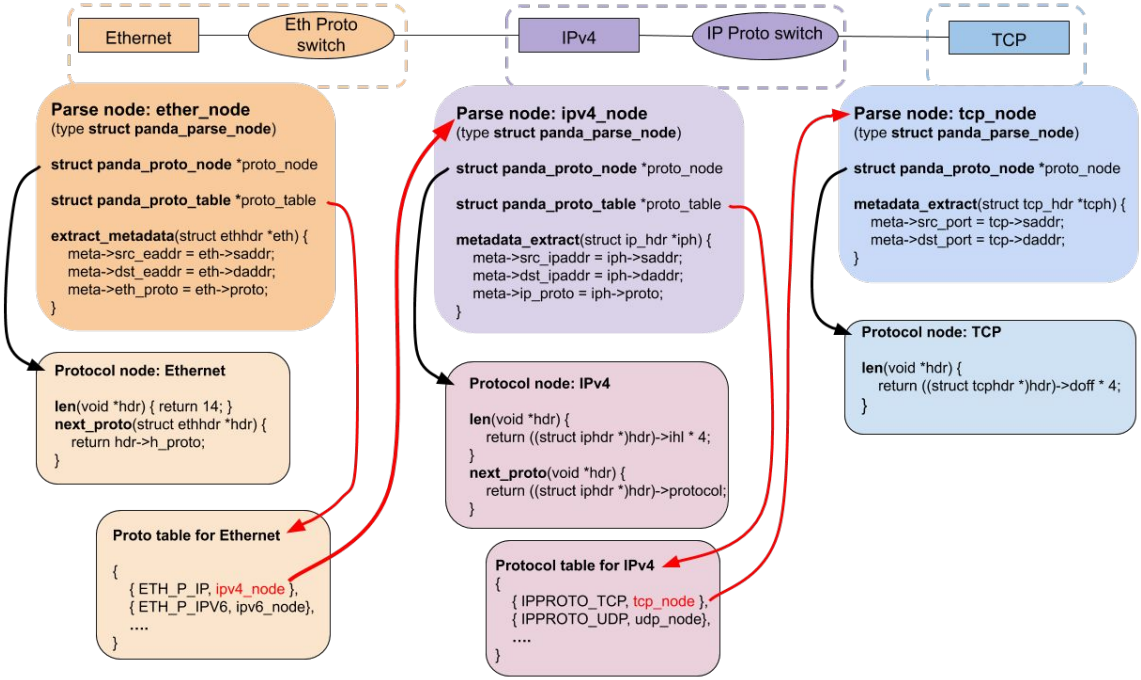
The PANDA Parser

- A fully programmable parser with a C/C++ frontend
  - Using C/C++ allows easy integration with rest of the stack
  - Other front end languages are possible
- A parser is defined by a declarative representation
  - A set of data structures linked together into a parse graph
  - Nodes in the graph are annotated with operations for parsing and metadata extraction
- Arbitrary backend software and hardware targets
  - Motto: *Write once, run anywhere, run well*
  - Same exact code runs across targets
- Additional features
  - Supports TLVs and flag-fields as first class citizens
  - Metadata frames
- **panda-compiler**: translates target-agnostic front end code into optimized executable

“Hop”

Programming model

- A parser is composed of a set of nodes linked together as a parse graph
- Node of the parse graph are the protocols to parse
- Each node contains protocol operations for parsing to
  - Length of current header
  - Next protocol
- Callbacks in each node allow for metadata extraction and custom processing of protocols
- Tables map protocol numbers to next nodes



Development path

Datapath program in C/C++

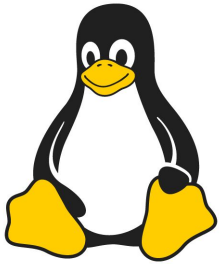
```
PANDA_MAKE_PARSE_NODE(ether_node,
    panda_parse_ether, NULL, NULL,
    ether_table)
...
```

Other language front ends
Python, Rust, P4, etc.
XML, JSON also!

panda-compiler

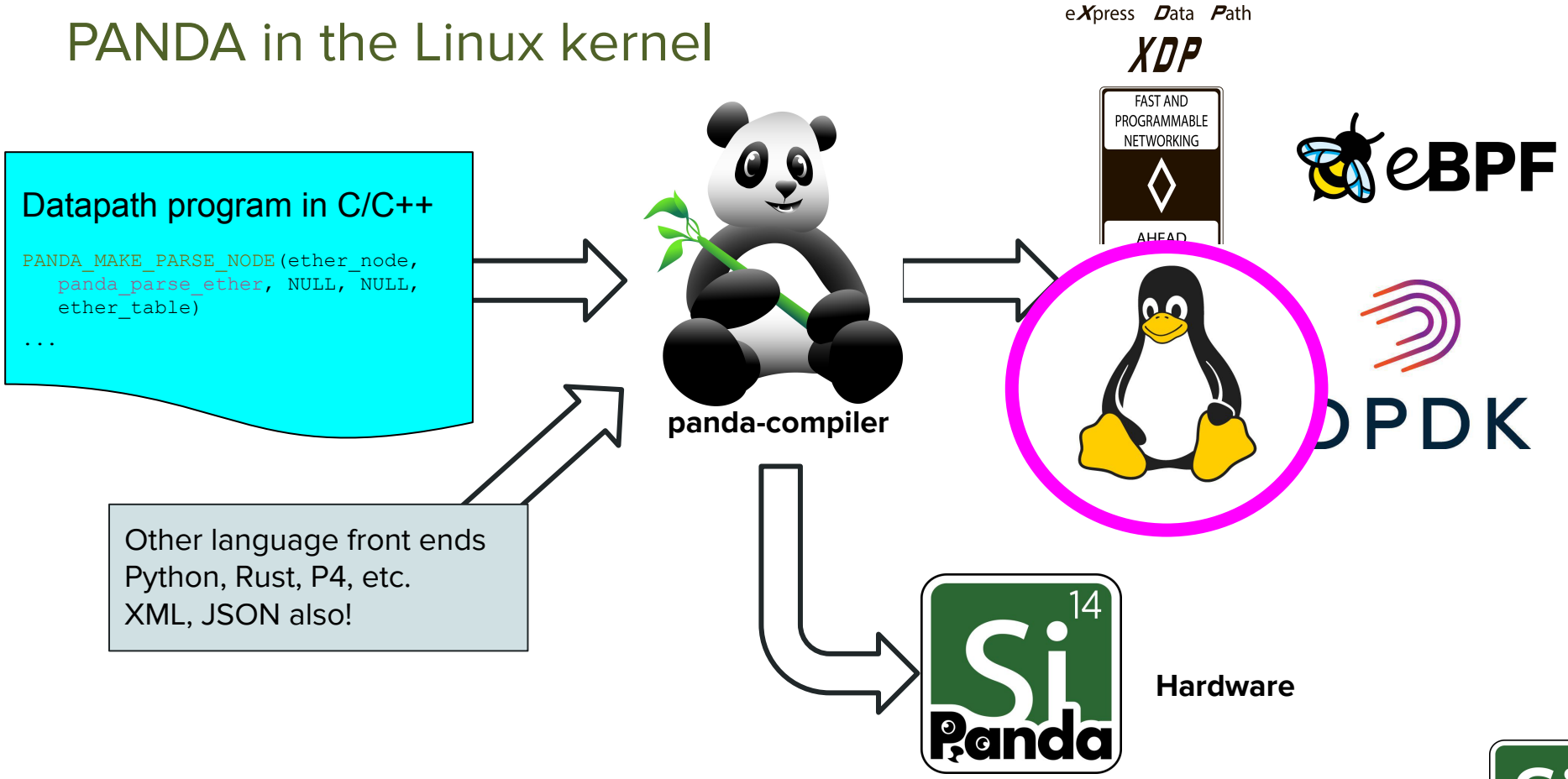
eXpress Data Path
XDP

FAST AND PROGRAMMABLE NETWORKING
AHEAD



Hardware

PANDA in the Linux kernel



Parser program (e.g. bigparser.c)

```
PANDA_PARSER_KMOD(
    panda_parser_big_ether, "",&ether_node,
        big_ether_panda_parse_ether_node
    );
/* Parser nodes and definitions */
...
```

panda-compiler

Optimized, kernelize C (e.g. parser.kmod.c)

```
 /* Inlined parser functions, unrolled paths */
...
static inline int panda_parser_big_ether(
     const struct panda_parser *parser,
     const struct panda_parse_node *parse_node,
     const void *hdr, size_t len,
     struct panda_metadata *metadata,
     unsigned int flags, unsigned int max_encaps) {
...
}
```

Kernel program (e.g. module.c)

```
/* Module glue, program logic, TC glue */
err = panda_parse(PANDA_PARSER_KMOD_NAME(
                  panda_parser_big_ether),
        pkt, pktlen,
       &mdata.panda_data, 0, 1);
...
PANDA_TC_MAKE_PARSER_PROGRAM("big",
                             do_parse);
```

gcc, link

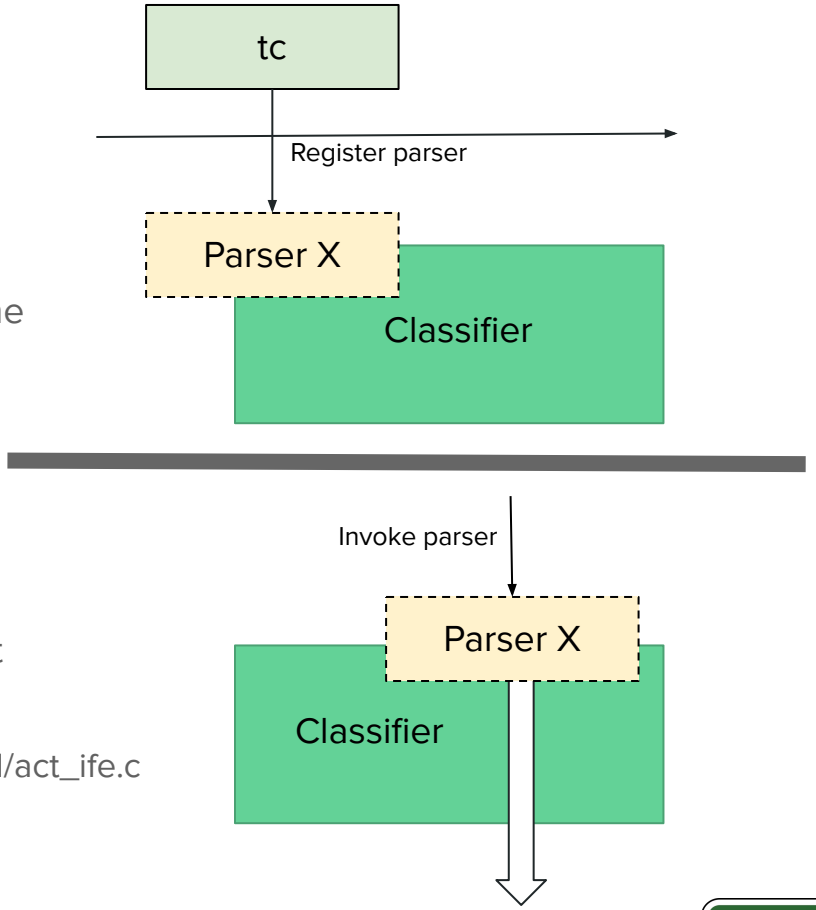
Loadable module .ko
(e.g. panda_big.ko)

modprobe

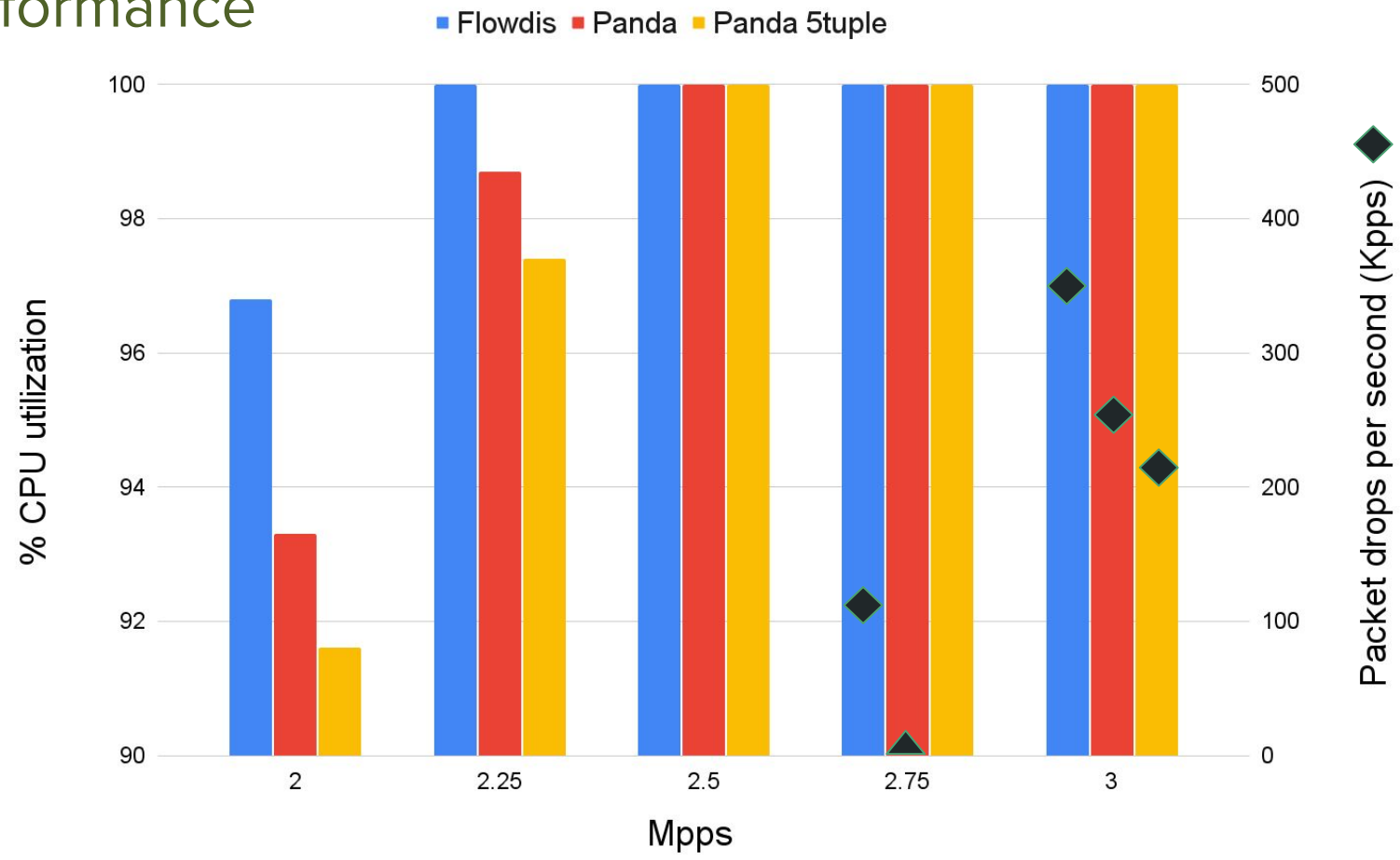
Building a module

The PANDA classifier

- Register the parser to the classifier
  - Doable via an argument on to the **tc** command line
  - tc filter [...] panda parser x [...]
- Associate the parser X to the filter Y
  - Once the parser is visible, this is trivially done
  - Store a struct ops to be invoked later
- Invoke parser X on classification
  - From the struct ops, invoke the parser entry point
- Inspirations drawn from act_ife
  - https://elixir.bootlin.com/linux/latest/source/net/sched/act_ife.c



Performance



Futures

- Metadata in BTF format
- Generic tc-flower
- Sunset flow dissector entirely
- Hardware acceleration of PANDA parser
- Programmable parser is not used in core stack protocol processing (hmm, maybe it could be!)

PANDA open source info

- https://github.com/panda-net/panda
- Releases
  - 1.0 - Introduce PANDA Parser
  - 1.1 - panda-compiler
  - 1.2 - PANDA Parser in XDP/eBPF
  - 1.3 - PANDA parser in kernel and the PANDA classifier (July 20, 2021)
- Inquiries: panda@sipanda.io

Thank you!

Q/A