Compilers are the future of Networking

Felipe Magno de Almeida
Netdev0x17

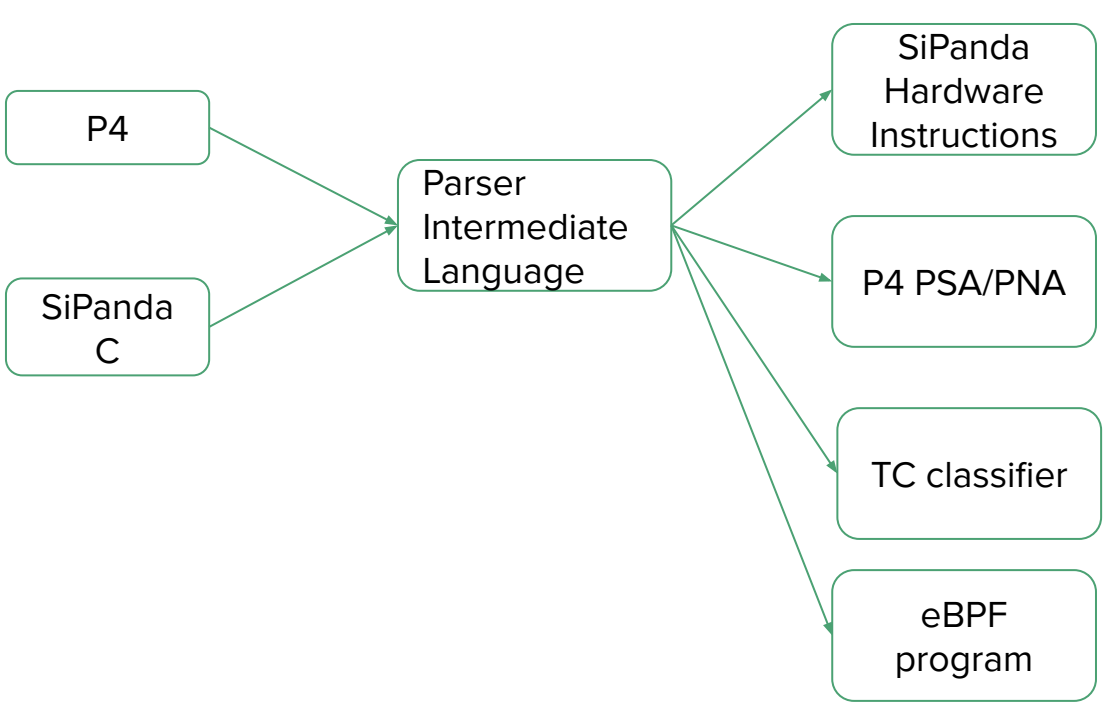
Compilers are optimization wizards

- Packet classification and packet handling is not simple anymore
- Optimizations are too complex for handwritten code
- Compiler optimizations feel magical sometimes
- Imperative C code is not appropriate for packet classification abstractions and concepts

Languages and parser abstractions

- P4 language
- SiPanda Declarative Framework for C language
- Parser Intermediate Language (JSON)
- Parser MLIR
- LLVM IR

Frontend Compilers and Abstract translations



Frontend SiPanda Compiler and Pattern Matching

```
%2 = load i8, i8* %0, align 4, !dbg !539
%3 = shl i8 %2, 2, !dbg !545
%4 = and i8 %3, 60, !dbg !545
%5 = zext i8 %4 to i64
ret i64 %5, !dbg !546
```

```
"name": "ipv4_node",
"min_len": 20,
"hdr_length": {
        "offset": 0,
        "mask": "0xf",
        "length": 1,
        "multiplier": 4
},
...
```

P4 Backend to Parser Intermediate Language

```
state parse_ipv4 {

packet.extract(headers.ipv4);

transition select(headers.ipv4.protocol) {
        PROTO_TCP: parse_tcp;
        PROTO_UDP: parse_udp;
        default: accept;
}
}
```

```
"name" : "parse_ipv4",
"min-hdr-length" : 20,
"next-proto" : {
        "field-off" : 9,
        "field-len" : 1,
        "table" : "parse_ipv4_table"
},
"metadata" : {
"ents" : [
{
            "name" : "headers.ipv4",
            "type" : "extract",
            "md-off" : 68,
            "hdr-src-off" : 0,
            "length" : 20
...
```

SiPanda compiler backend

- Leverages on MLIR and LLVM IR
- Generates Hardware Accelerated Instructions or eBPF and even P4
- Leverages on LLVM optimization passes
- Can be easily extended for other Hardware Accelerated Devices

Thank You!