JIT compilation for packet filtering using Netmap and LLVM

Daniel Peyrolón
David Chisnall

Sep 16, 2014

Purpose of the project

- This is a GSoC project for FreeBSD.
- Development effort.
- Leverage Netmap for quick packet filtering.

Netmap

- Kernel module.
- Maps NIC rings with userspace.
- Userspace network stack.
- High speed for network operations.

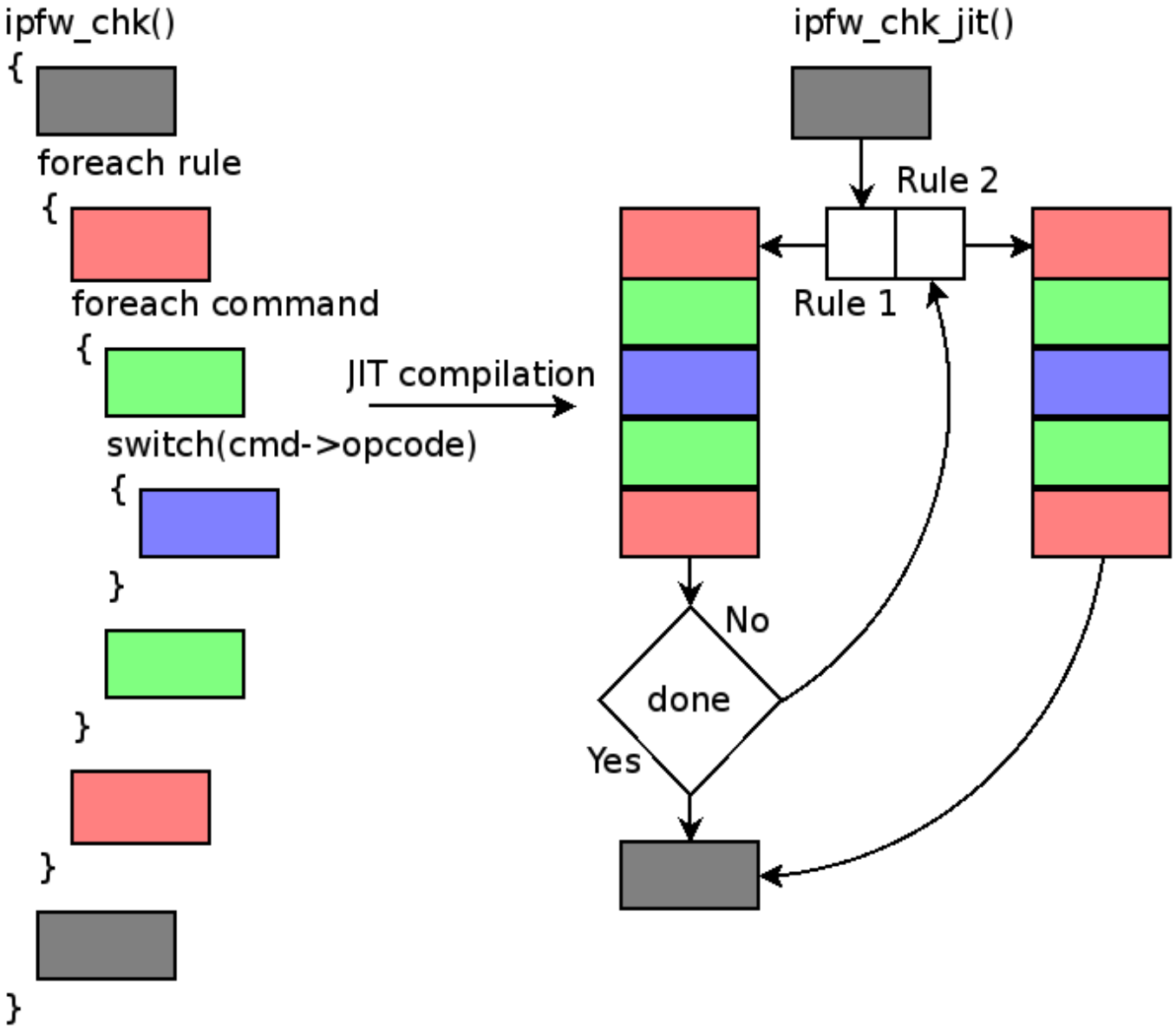
LLVM

- Compiler framework.
- Program compilation, analysis and transformation.
- Widely used.

How does it work?

- How rules are interpreted.
  - Example: *"accept tcp from any to any 80"*
- How the JIT works.
  - Loads external LLVM bitcode.
  - Bitcode contains functions and structs used.
  - Functions called from the JIT, inlined.
  - Iterate through the ruleset and emit code.

Compilation



Benefits from this approach

- It's easy to develop and update the compiler.
- General solution for packet filtering.

```
 case O_ACCEPT:
    rule_accept(&retval, &l, &done );
    break;

emit_accept(){
    Irb.CreateCall(RuleAccept, {Retval, L, Done});
}
```

Basic benchmarking

Basic benchmark for 1k pkts

JIT compiler	Compilation	130ms
	Filtering	**523 μs**
Interpreter	Filtering	3664 μs

Speedup = x7 for filtering code

Compilation time $\equiv$ Interpreting rules for 35480 pkts

Packets needed for amortization $\equiv$ 41440 pkts

Future work

- Complete the firewall.
- Benchmarking, evaluation.
- Static analysis.
- Feedback-driven optimisations.

What I'm trying to say

- It works!
- Perhaps it's interesting for someone?
- Ongoing development effort.
- Thanks to many people.

Thanks for your attention!

Questions?
Suggestions?