OARC 28th Workshop
San Juan, Puerto Rico
8 March 2018

There is always time for DNS

Jerry Lundström
Software Engineer

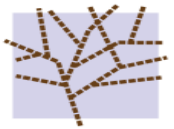
Background

- Started working for OARC some two years ago
- Saw a lot of code duplication, even more after acquiring dnscap
- Created helper libraries of the shared code as Git submodules
- The modules helped a lot when creating drool

But… problems...

Problems

- dnscap dynamic library plugins
  - Not trivial to create
  - Portability issues
  - ABI issues
  - /usr/lib vs /usr/lib64
  - Packaging issues
  - Third party libraries
  - Third party plugins?

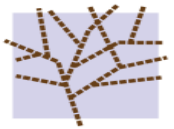


Problems

- drool configuration
  - One “scenario” is easy… multiple is hard
  - Configuration syntax might evolve into a language on its own
- Helper libraries… didn’t help in some cases
  - A tiny fix might result in several releases (okay, that’s not really the libraries fault but anyway...)

Looking for the Solution

- If drool's configuration syntax would eventually evolve into a language… why not just use a language that already exists and extend it!
  - Would solve drool's multiple scenarios issue
  - Would solve dnscap plugins creation, distribution etc
  - Should be possible to rewrite DSC dataset into scripts
  - Code snips could work as well as PacketQ

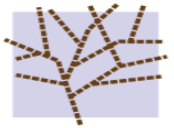


Looking for the “Right” Solution

- Embeddable script engines
  - Perl… NO
  - Python… no
  - Lisp… (((no)))
  - JS… didn’t actually check
  - Lua… oh hey, this was made to embed!
  - LuaJIT… oh wow, that is fast!

Why Lua / LuaJIT

- Made to be embedded
- Have been around for a while
- Already embedded in other DNS software
  - PowerDNS, dnsdist, Knot Resolver…
- Used in other high performance network frameworks
  - snabb, libmoon



Why not use snabb / libmoon?

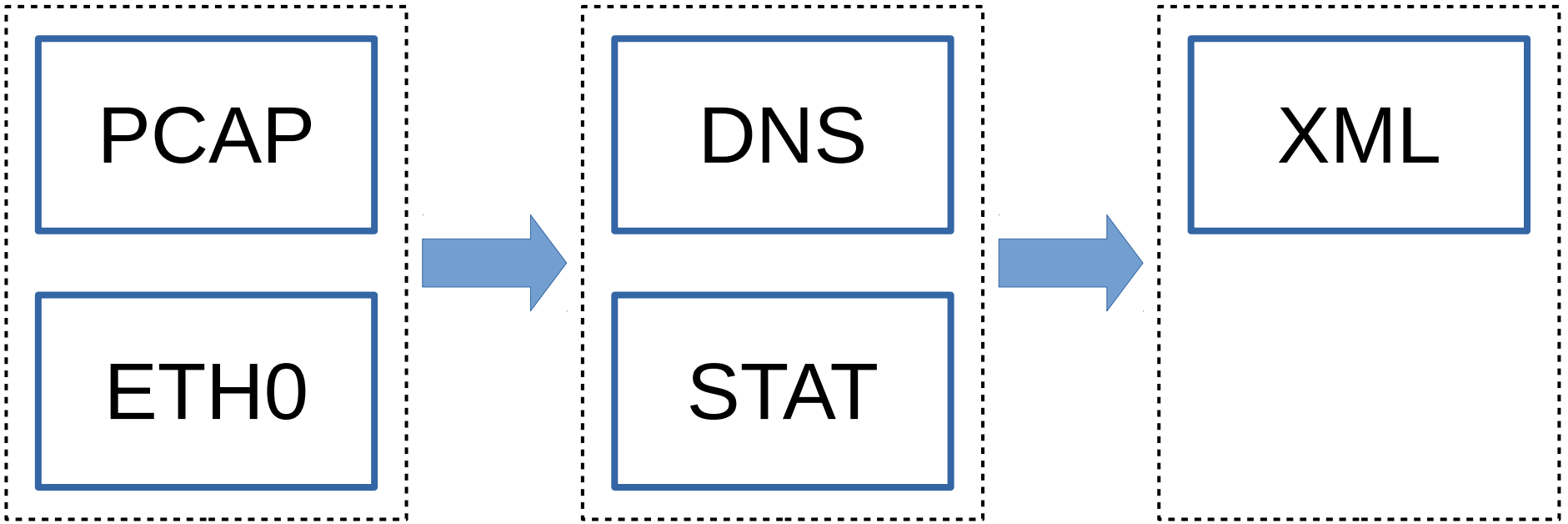
- I’ll get to that in a bit… but first

Existing Features

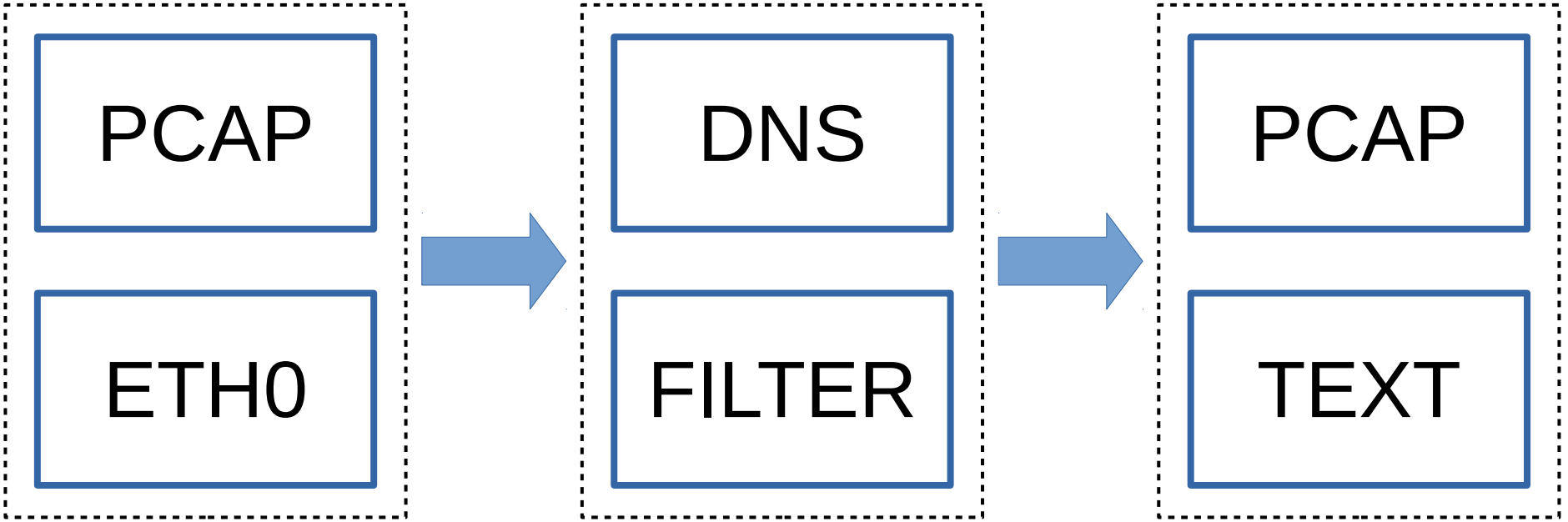
The Tools

- dsc – DNS Statistic Collector
- dnscap – DNS Capture
- drool – DNS Replay Tool
- packetq – Basic SQL-frontend for PCAPs

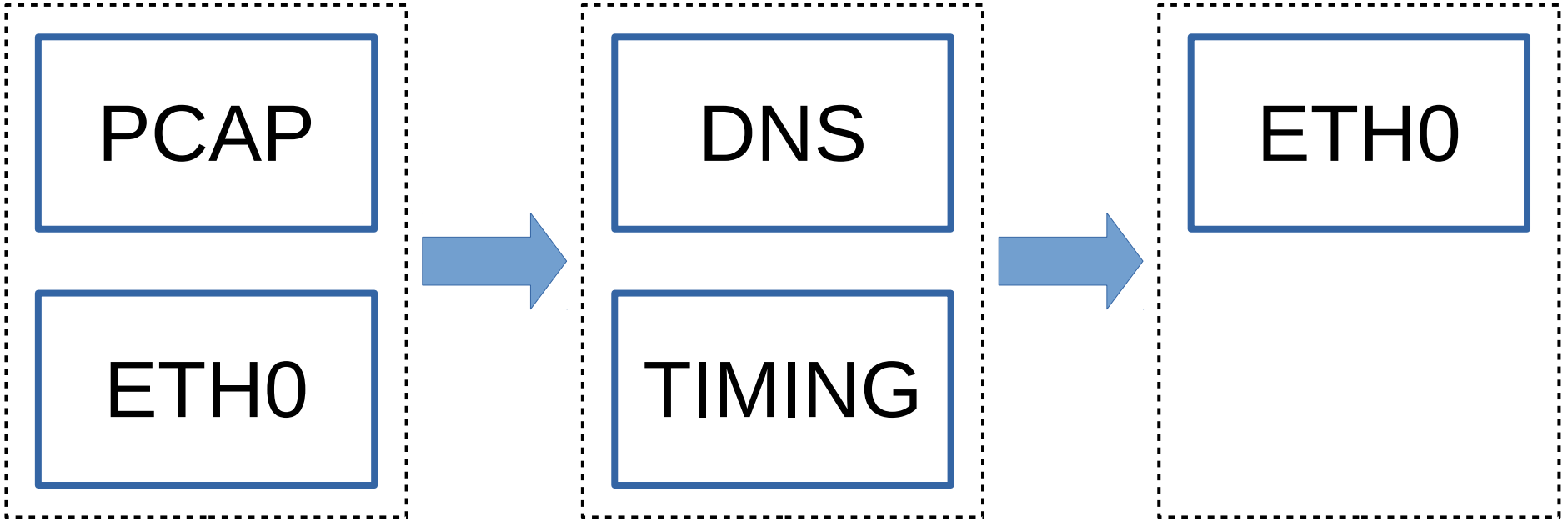
dsc – DNS Statistic Collector



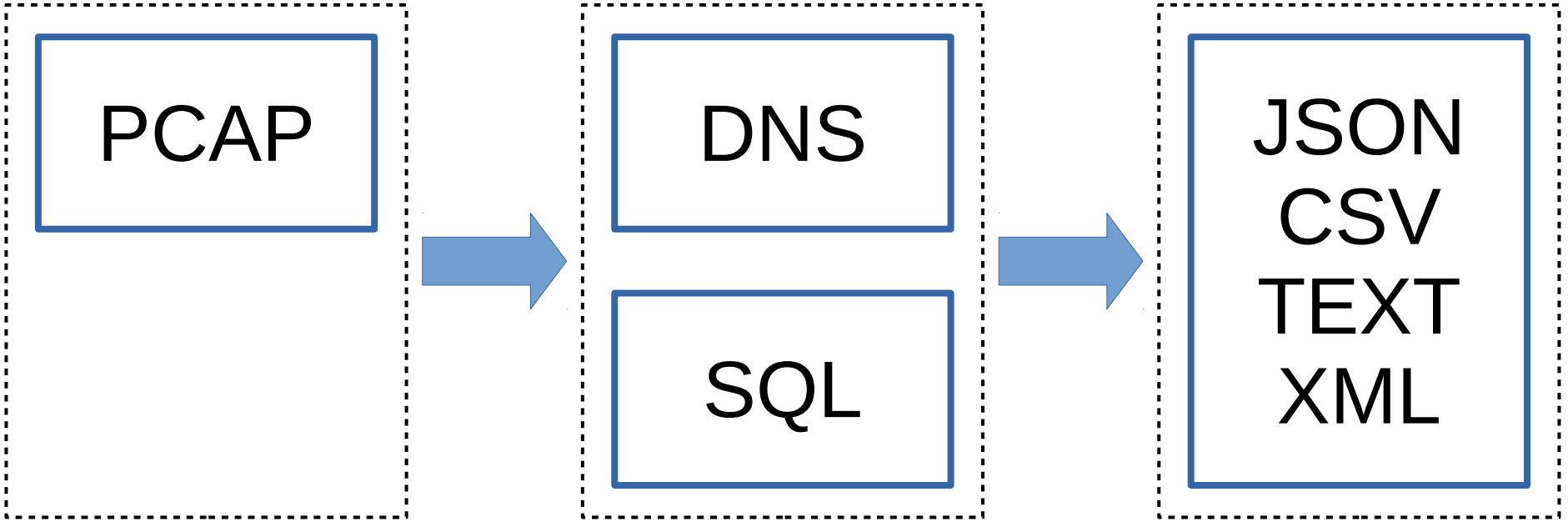
dnscap – DNS Capture



drool – DNS Replay Tool

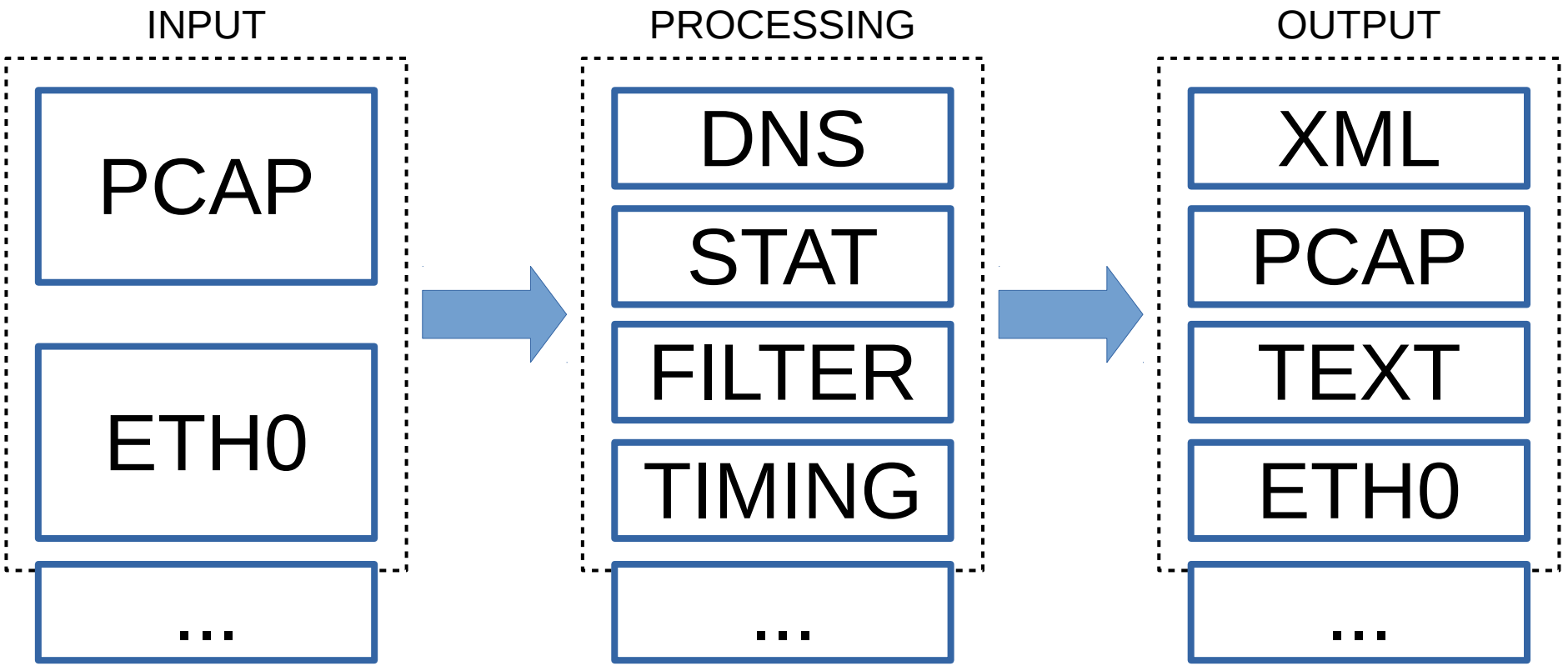


packetq – SQL-frontend for PCAPs



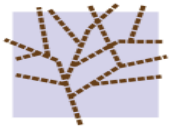
Hmm…

The One Tools



Lua Test Run

- Took some time to test it out
- Chopped up drool
- In two days had put it together with Lua standard interface… and it worked!
- Decided to continue and refactor into LuaJIT FFI, all functionality taken from drool
  - This is why not snabb / libmoon, but it does not prevent using them later on



dnsjit

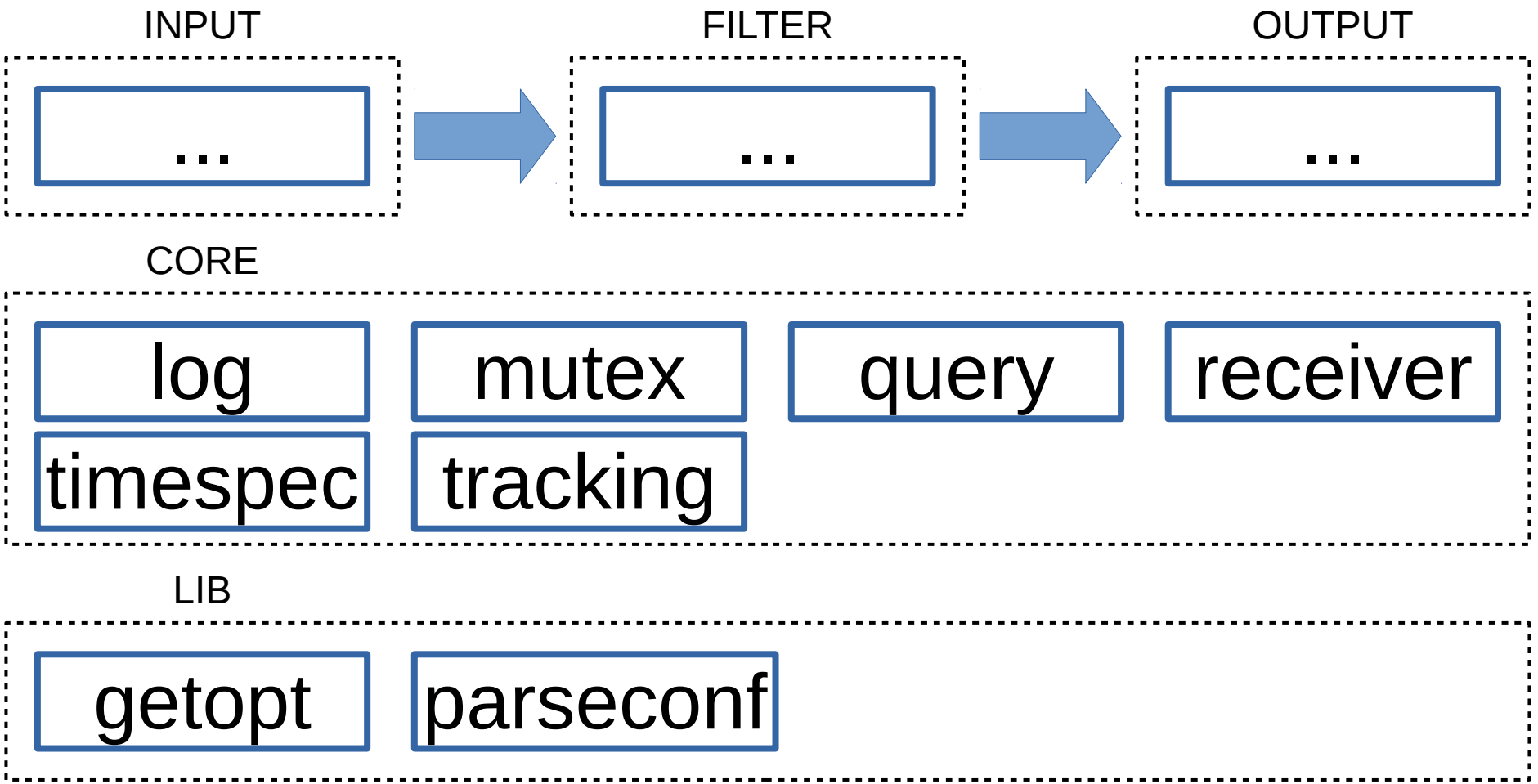
https://github.com/DNS-OARC/dnsjit

“dnsjit is a combination of parts taken from dsc, dnscap, drool, and put together around Lua to create a script-based engine for easy capturing, parsing and statistics gathering of DNS message while also providing facilities for replaying DNS traffic.”

come by the demo booth

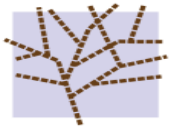
Design

dnsjit


INPUT
...
FILTER
...
OUTPUT
...
CORE
log
mutex
query
receiver
timespec
tracking
LIB
getopt
parseconf

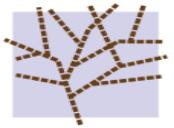
dnsjit

- input
  - lua
  - pcap
  - zero
- filter
  - lua
  - multicopy
  - roundrobin
  - thread
  - timing
- output
  - cpool
  - null



dnsjit.core

- log: Logging configuration and functions that can be applied globally, per module or per instance
- mutex: For any kind of action that requires global locking
- query: The main structure that is passed around, holds raw packet, network/protocol information and parsed DNS



dnsjit.core

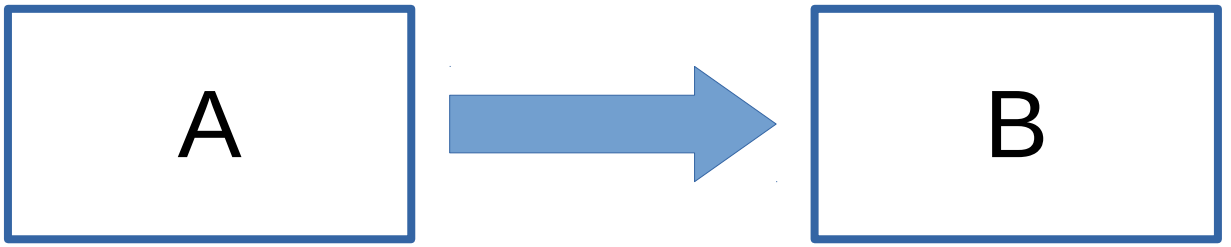
- receiver: Defines the receiver()/receive() interface
- timespec: As *struct timespec* but arch independent
- tracking: Global thread safe counters for tracking query and responses
  - Source ID (src_id), Query/Response ID (qr_id), Destination ID (dst_id)

dnsjit.lib

- getopt: A getopt()/getopt_long()-like Lua implementation
- parseconf: Reimplementation in Lua of the helper library parseconf, used in DSC and drool

Usage

The Chain

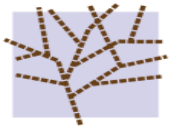


`receiver()` `receive()`

```
a:receiver(b)
func, obj = b:receive()
```

Example

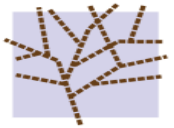
```
local input = require("dnsjit.input.pcap").new()
local output = require("dnsjit.filter.lua").new()

output:func(function(filter, query)
    query:parse()
    print(query.id)
end)

input:open_offline("file.pcap")
input:receiver(output)
input:run()
```



DNS-OARC
Domain Name System Operations Analysis and Research Center

Example

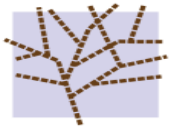
```
local input = require("dnsjit.input.pcap").new()
local filter = require("dnsjit.filter.timing").new()
local output = require("dnsjit.output.cpool").new(
  "127.0.0.1", "53"
)
input:open_offline("file.pcap")
input:receiver(filter)
filter:receiver(output)
output:start()
input:run()
output:stop()
```



DNS-OARC
Domain Name System Operations Analysis and Research Center

Documentation

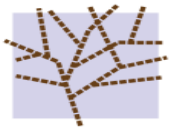
```
man dnsjit

man dnsjit.core

man dnsjit.lib

man dnsjit.input

man dnsjit.filter

man dnsjit.output

man dnsjit....
```



DNS-OARC
Domain Name System Operations Analysis and Research Center

Current State

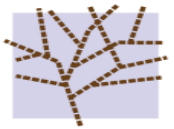
- Alpha v0.9.x
- No tar-balls made, use develop branch
- No tagging done, version bumped as packages are made
- Packages for most dists @ pkg.dns-oarc.net
  - Only missing RPM ppc64le support due to LuaJIT
- No tests yet
  - but drool develop branch tests works



DNS-OARC
Domain Name System Operations Analysis and Research Center

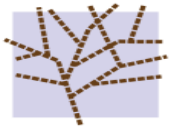
dnsjit v1.0.0

- Finalized the structure to pass around
  - query is a bit bloated today and slow to alloc/free
- DNS parsers
  - dnsjit.lib.omgdns
  - dnsjit.lib.ldns
- Rework dnsjit.filter.thread
- Ready for drool v2



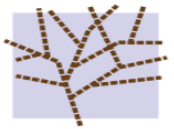
drool v2.0.0+

- Develop branch already rewritten in Lua
- Same options and config syntax
- Very close to the same performance (~98%)
  - But this is due to that dnsjit does a lot more then drool and the query structure still need a lot of work
- Comcast project; Performance and Responses
  - Configurable thread models, non-thread, atomic queues



dnscap v2.0.0+

- Rewrite to Lua/dnsjit
- Load Lua instead of dynamic libraries
- Rewrite all plugins to Lua
- Rewrite `-g` to Lua
- Rewrite BPF to Lua
- Pull TCP reassemble/ongoing into dnsjit



That’s a wrap, now go learn Lua!