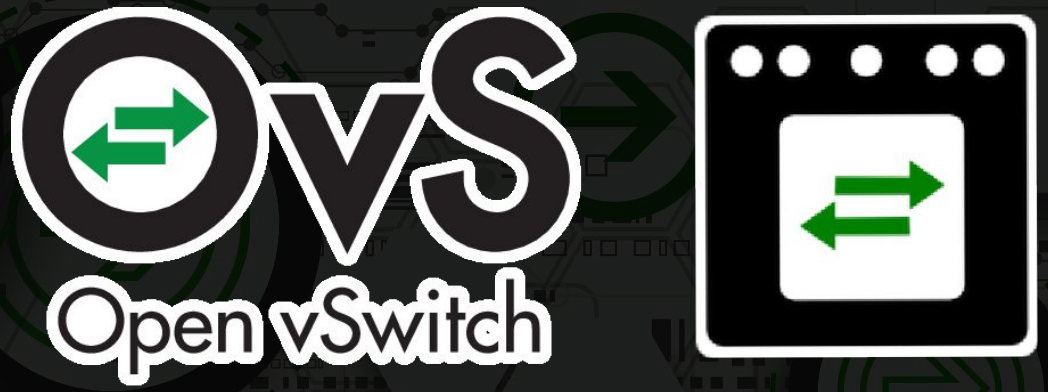


Prague, November 20-21, 2024

Low on Stack: Flexible protocols vs limited memory.

Ilya Maximets, Red Hat

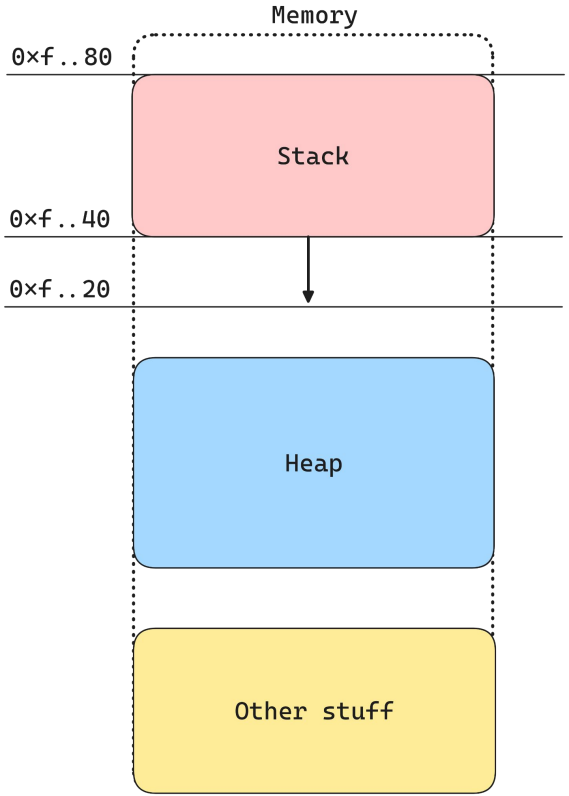
Stack: An efficient source of memory

Usually, just a single instruction to allocate a chunk of memory:

```
sub  $0x20,%rsp
```

Relatively limited space:

```
$ ulimit -s
 8192
```



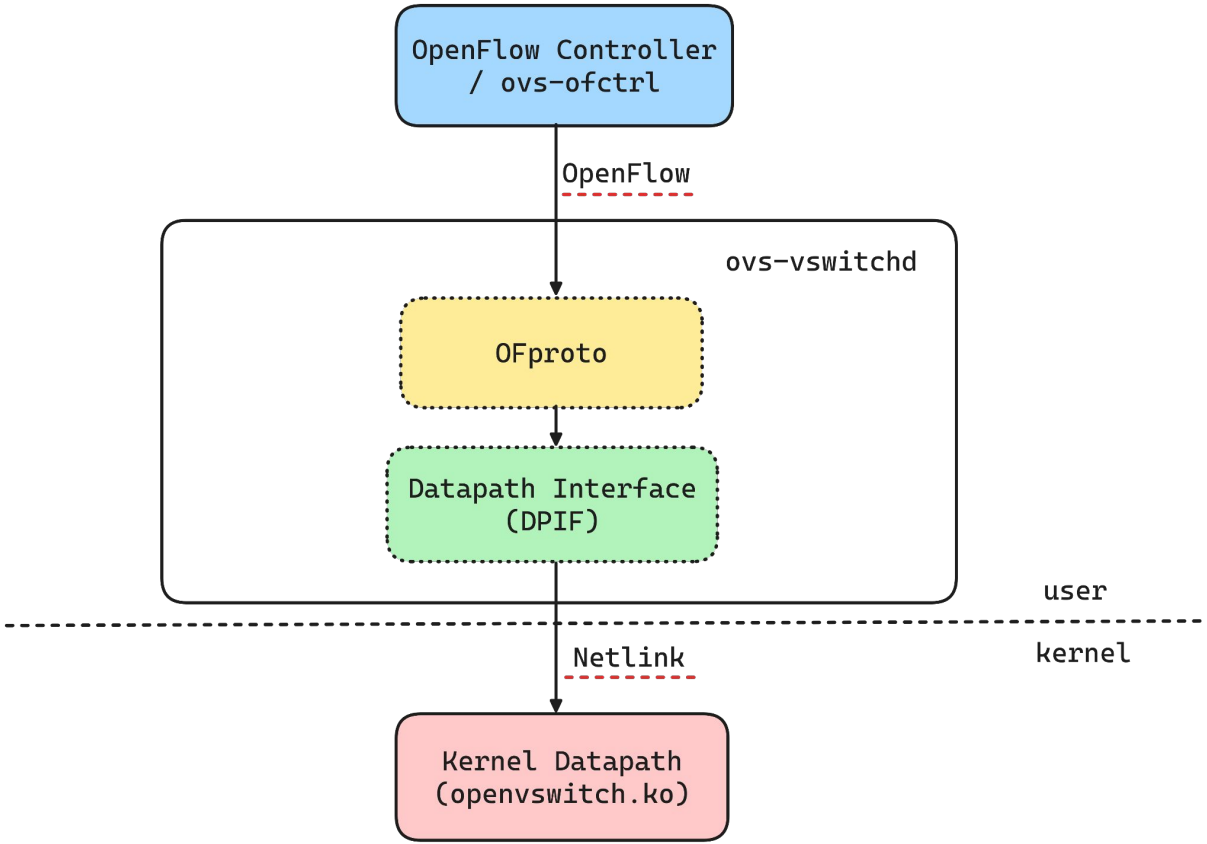
Stack usage in a typical function call

1. Push current context (registers that will be used later) onto stack.
2. Allocate stack space for local variables. (sub)
3. Do the work.
4. Deallocate the local stack space. (add)
5. Pop / restore the context back.
6. Return from the call

Compilers sometimes are evil!

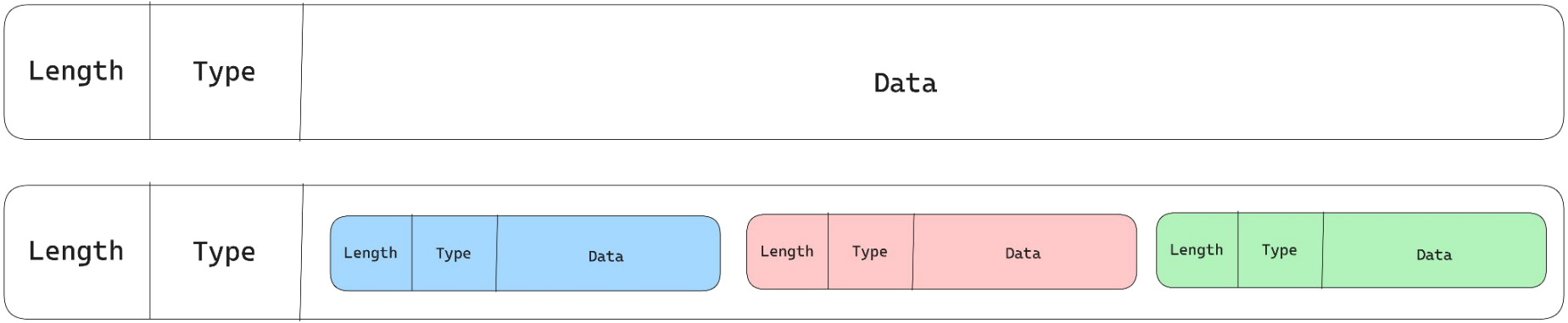
over-inlining of functions leads to increased stack usage!

Protocols in OVS

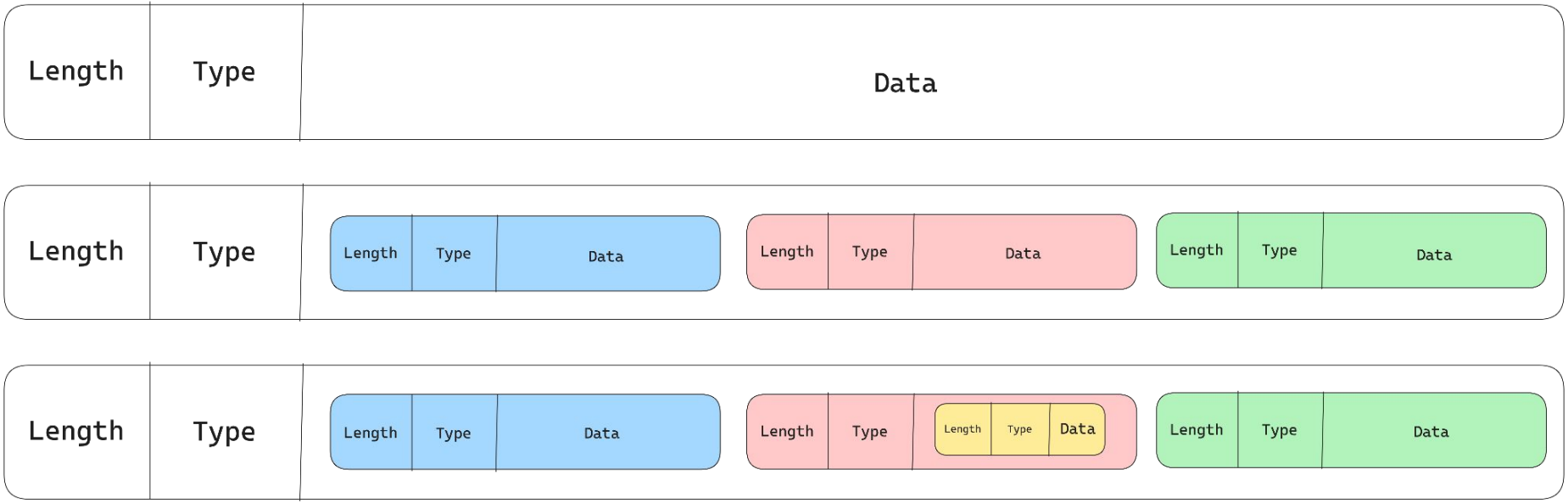


Netlink protocol

Netlink protocol

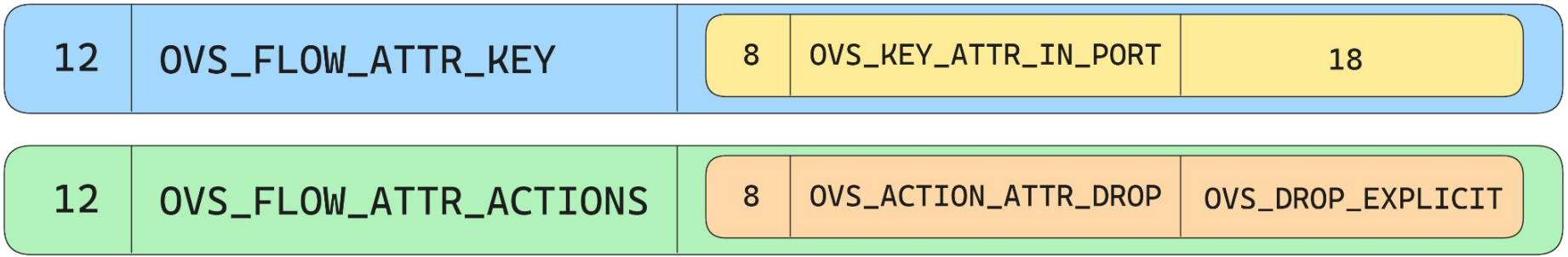


Netlink protocol



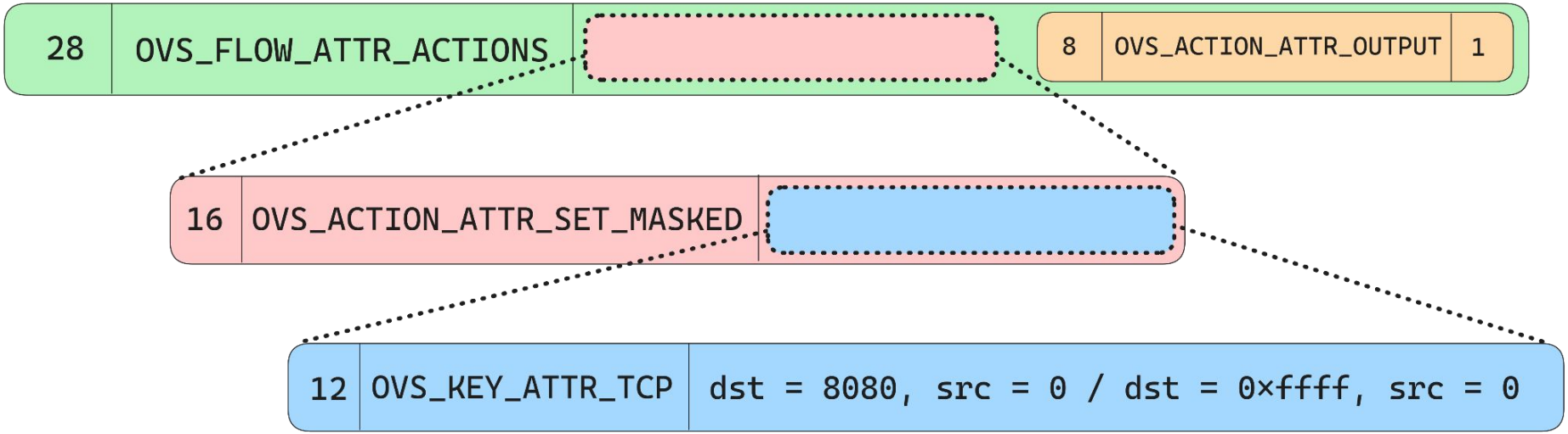
Netlink protocol

```
in_port(18), actions:drop
```



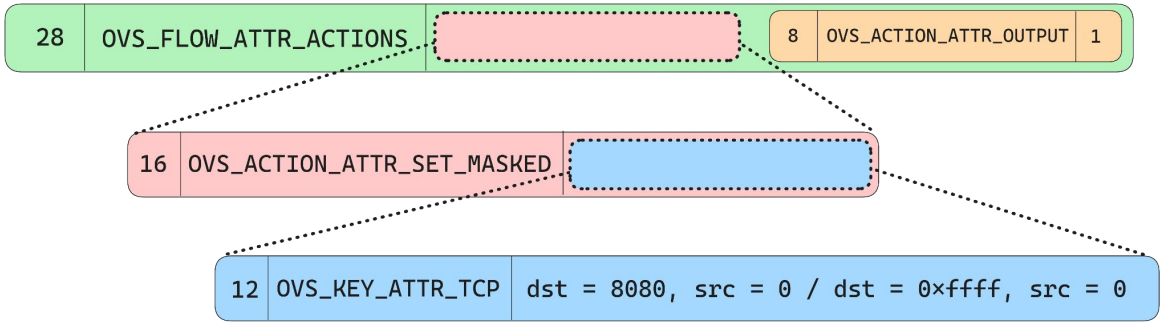
Netlink protocol

```
actions:set(tcp(dst=8080/0xffff)),1
```



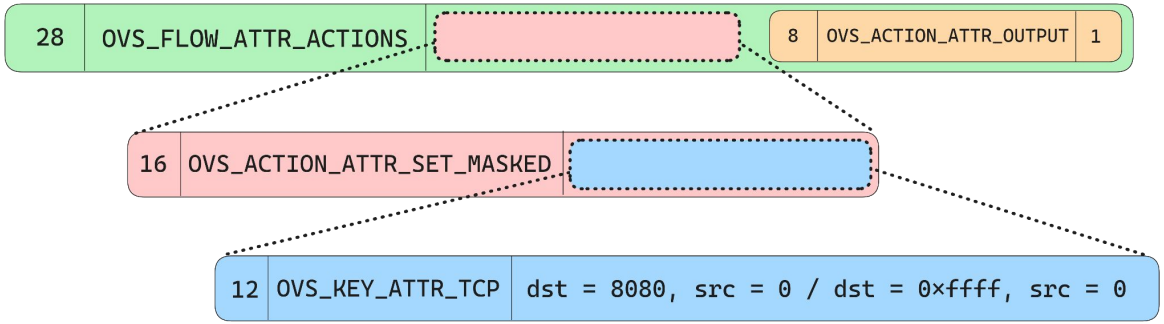
Netlink

- Parsing:
  - Validation
  - Enrichment
- Execution



Netlink

- Parsing:
  - Validation
  - Enrichment
- Execution



Done recursively, of course…

Netlink: Parsing

- Netlink attribute length - 16 bit (64kB max size)
  - Can fit a lot of nested attributes!
  - Flow keys do not allow a lot of nesting - not generally a problem.
  - But actions do: clone(clone(clone(clone(drop))))

Netlink: Parsing

- Netlink attribute length - 16 bit (64kB max size)
  - Can fit a lot of nested attributes!
  - Flow keys do not allow a lot of nesting.
  - But actions do: clone(clone(clone(clone(drop))))

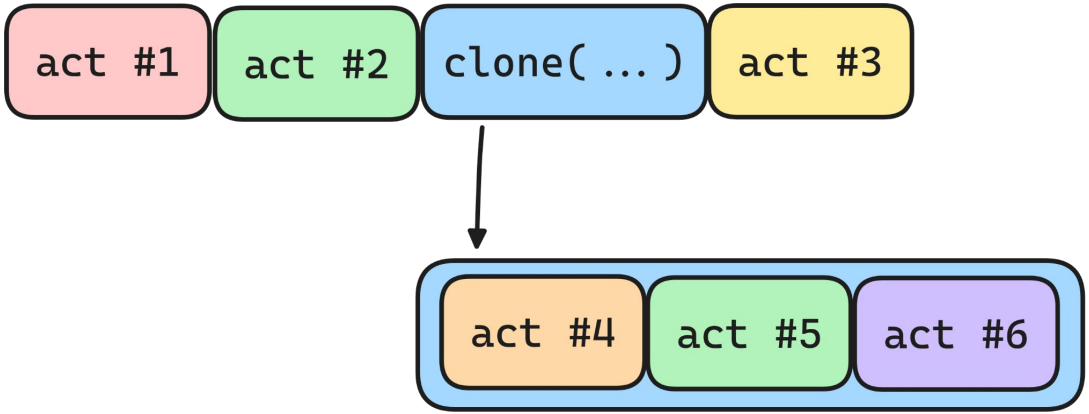
```
OVS_COPY_ACTIONS_MAX_DEPTH 16
```

Netlink: Execution

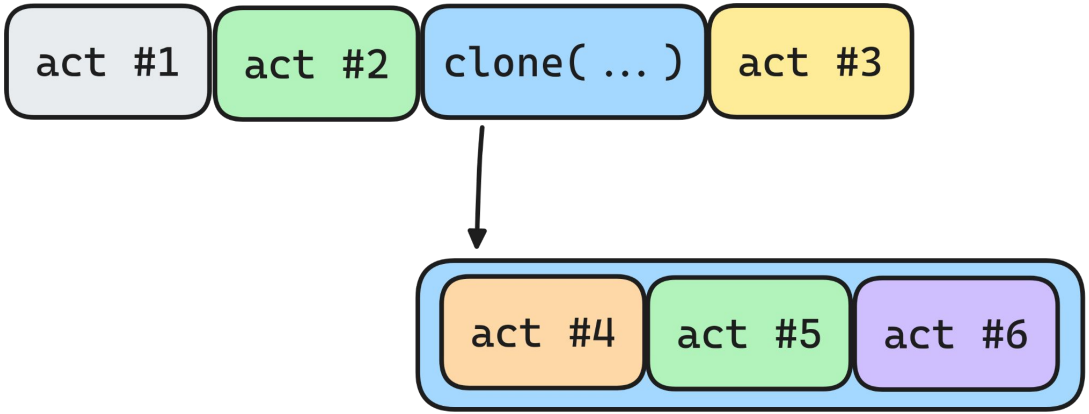
- Kernel stack size: ~ 16kB
  - Can’t go too deep into recursion.

- Should not spend a lot of time on one packet.

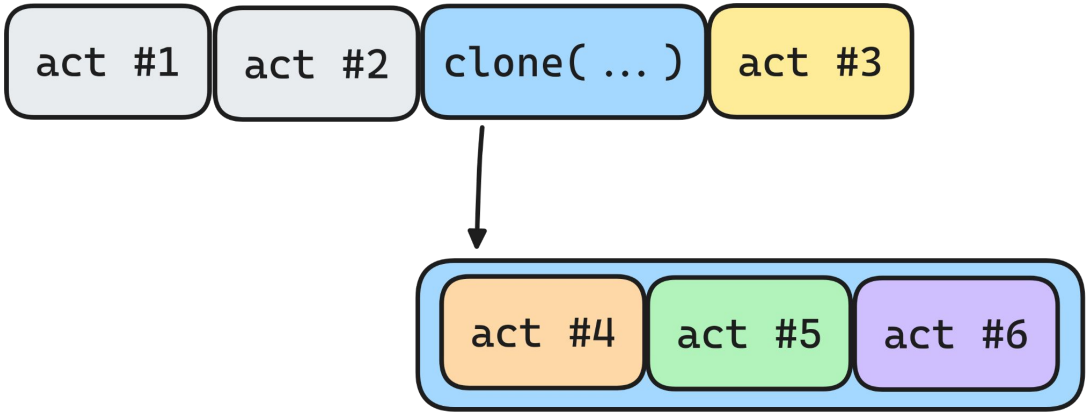
Netlink: Execution



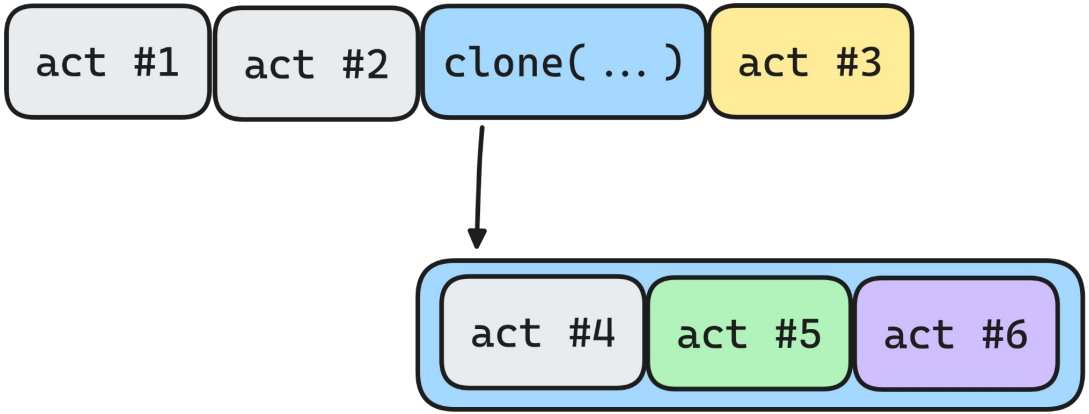
Netlink: Execution



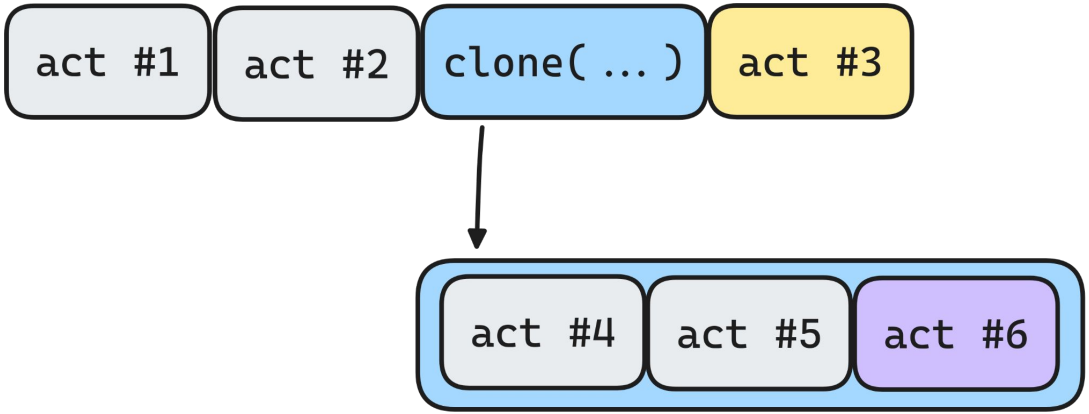
Netlink: Execution



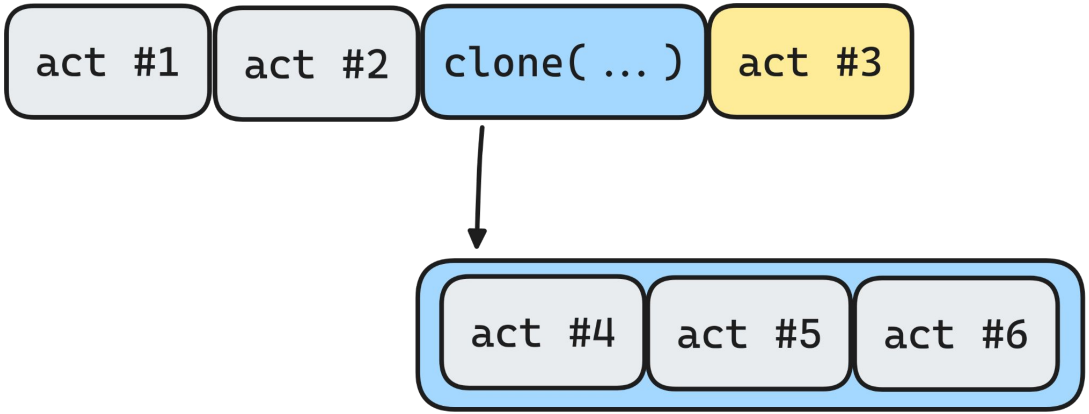
Netlink: Execution



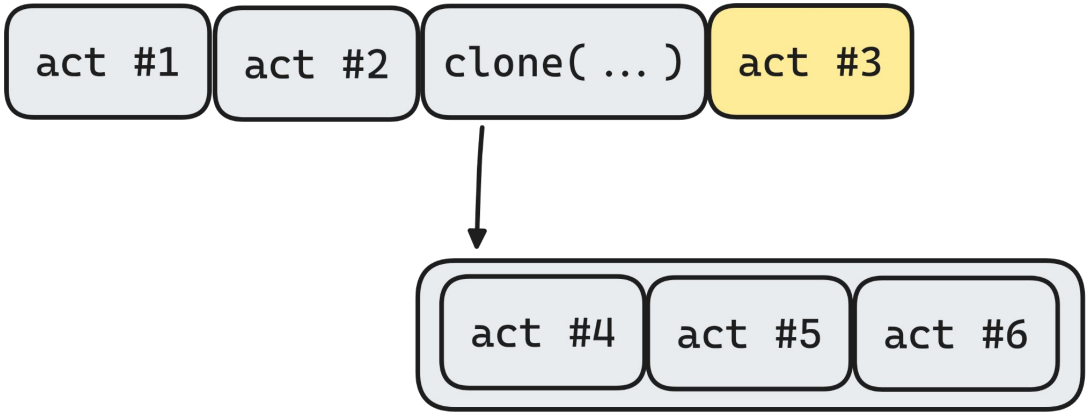
Netlink: Execution



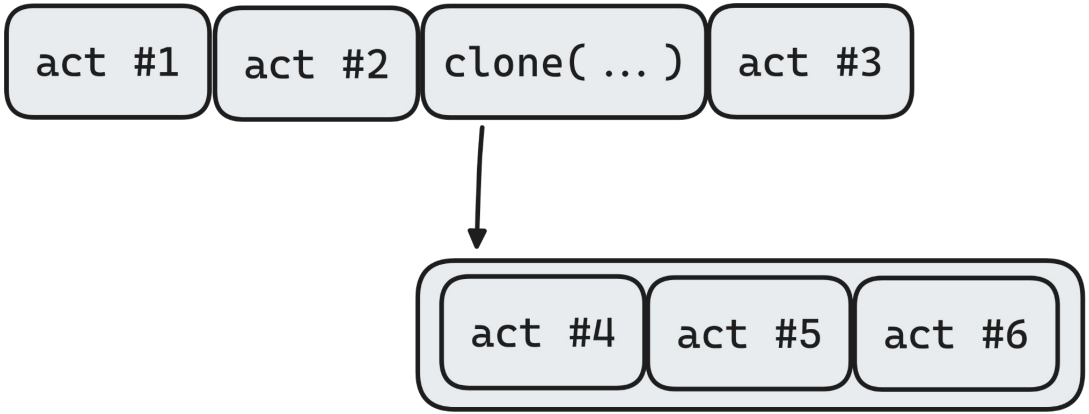
Netlink: Execution



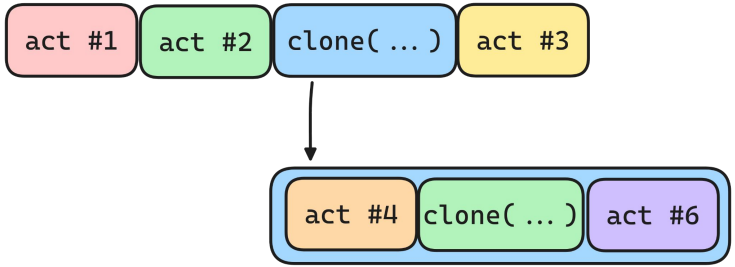
Netlink: Execution



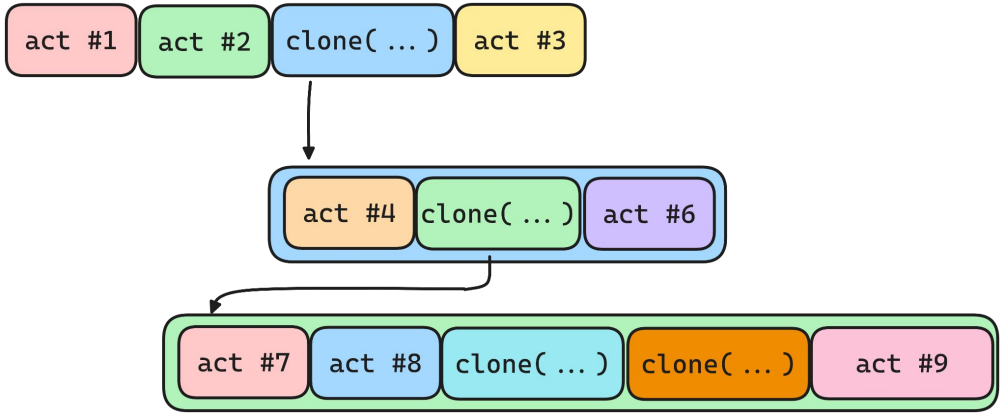
Netlink: Execution



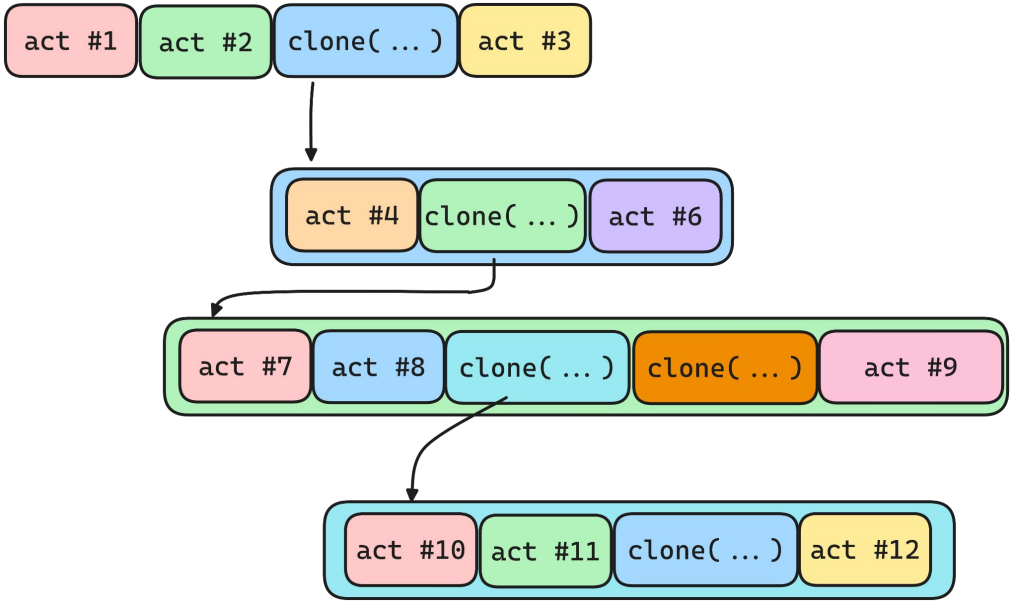
Netlink: Execution



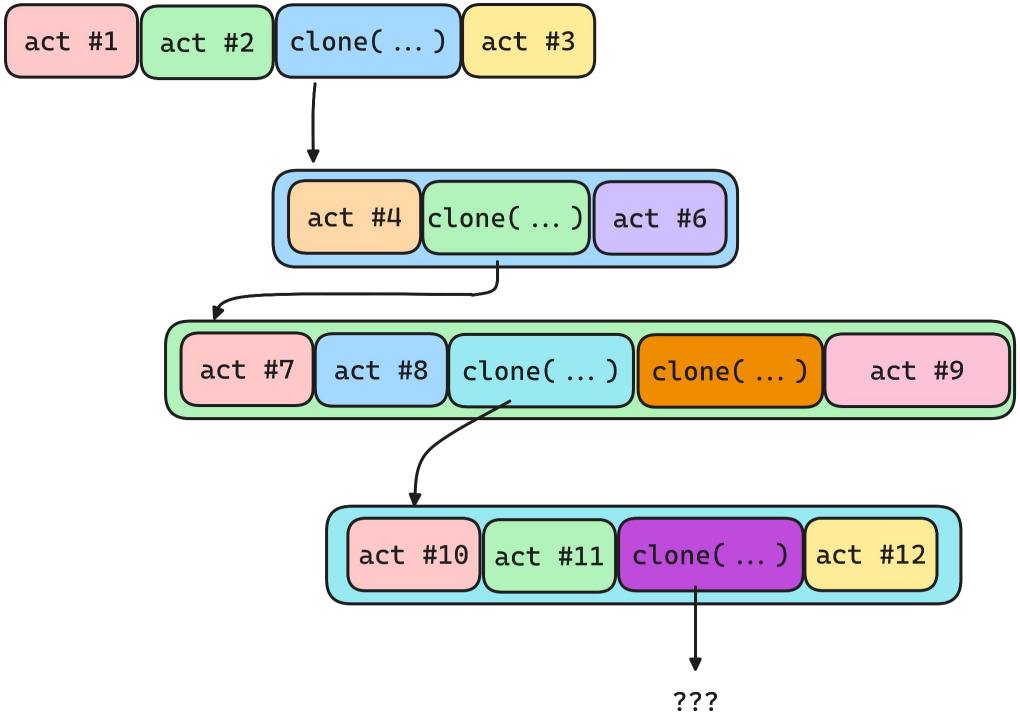
Netlink: Execution



Netlink: Execution

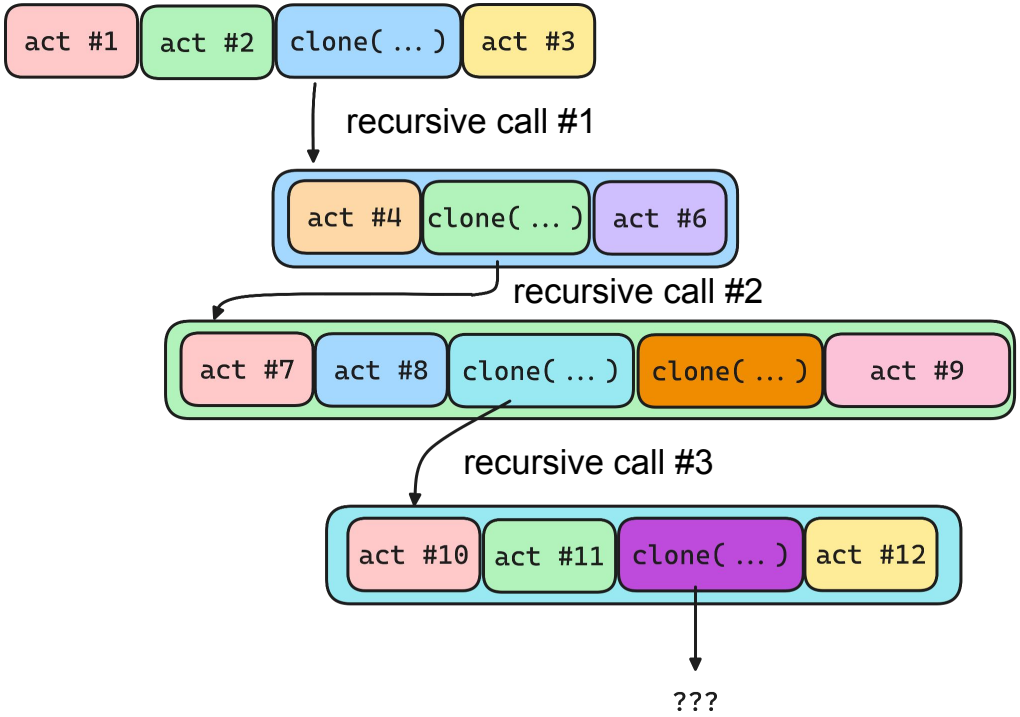


Netlink: Execution



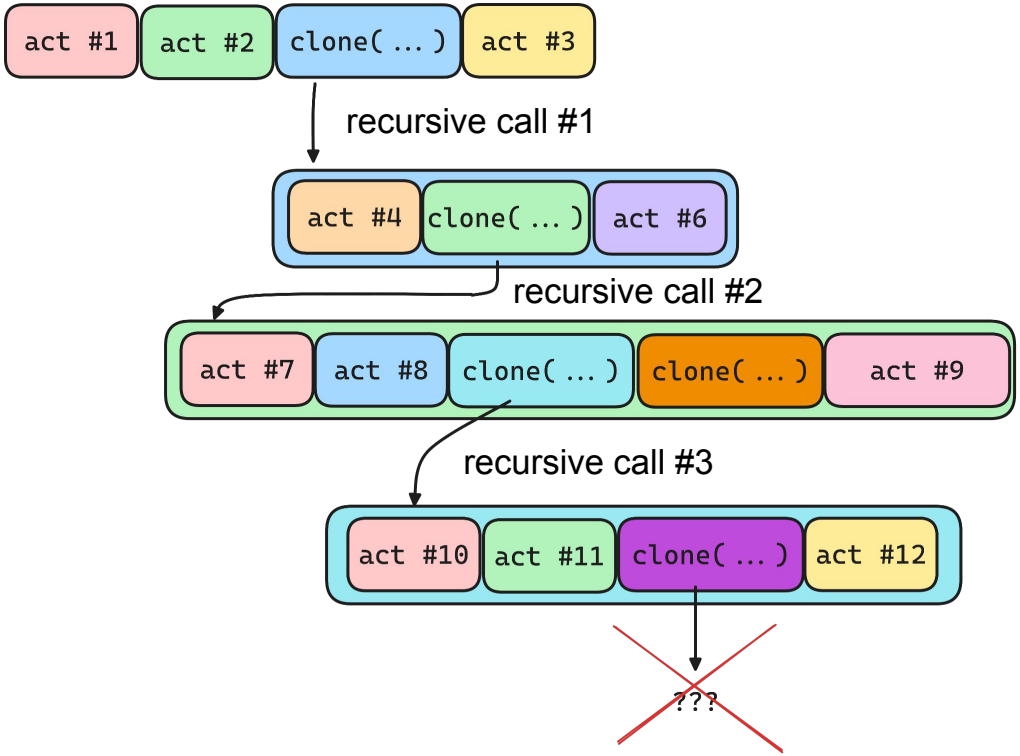

```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3
```

Netlink: Execution




```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3
```

Netlink: Execution

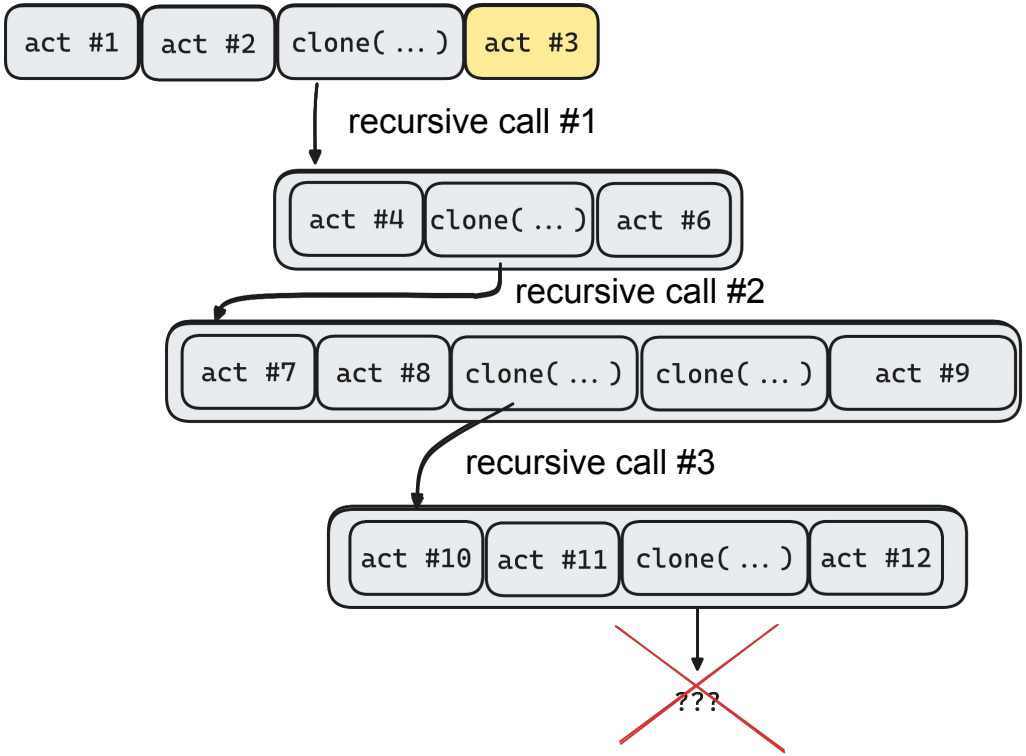



```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3
```

Deferred Actions FIFO:

1. act #13 act #14 act #15
2.
3.
4.
…
9.
10.

Netlink: Execution

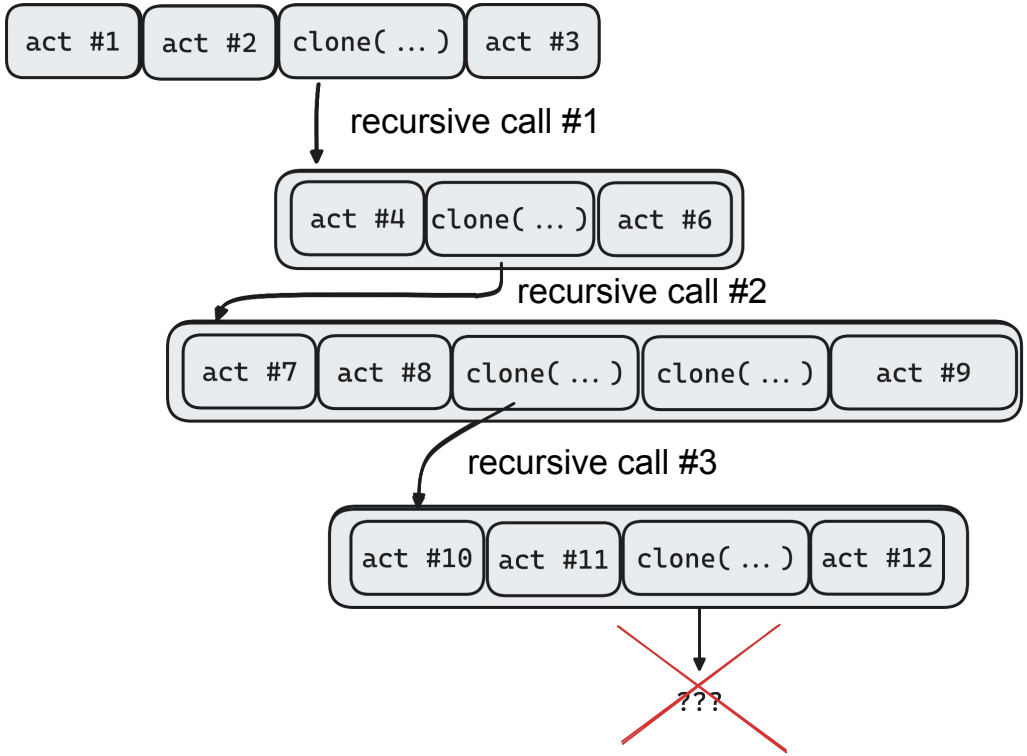



```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3
```

Deferred Actions FIFO:

1. act #13 | act #14 | act #15
2.
3.
4.
…
9.
10.

Netlink: Execution

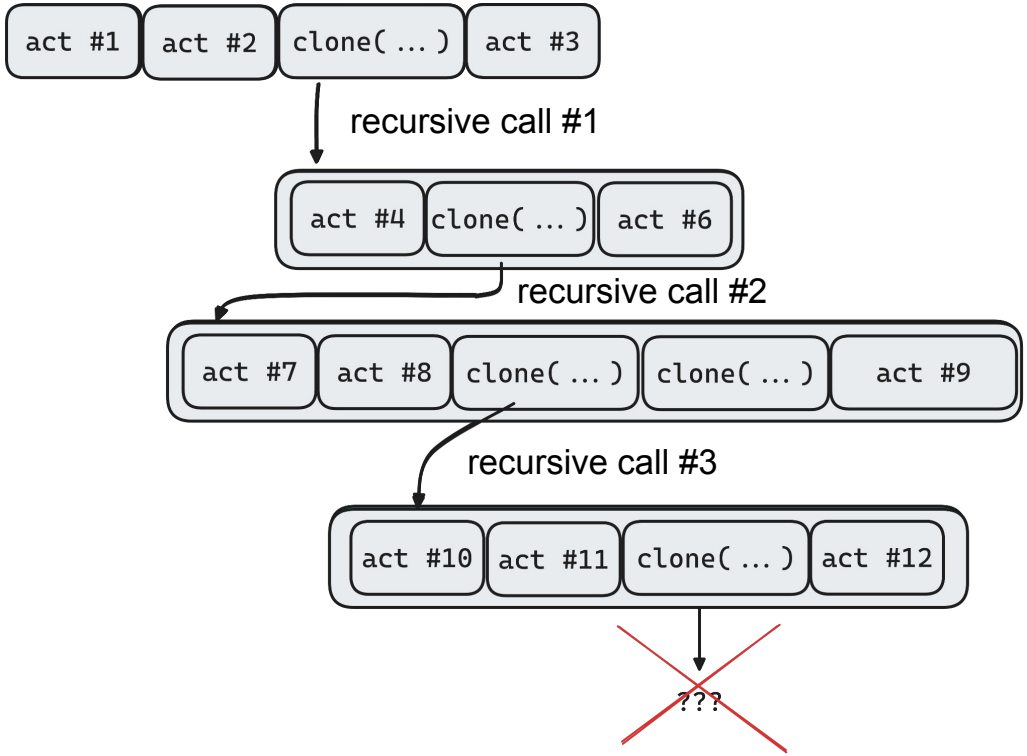



```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3
```

Deferred Actions FIFO:

1. act #13 | act #14 | act #15
2.
3.
4.
…
9.
10.

Netlink: Execution

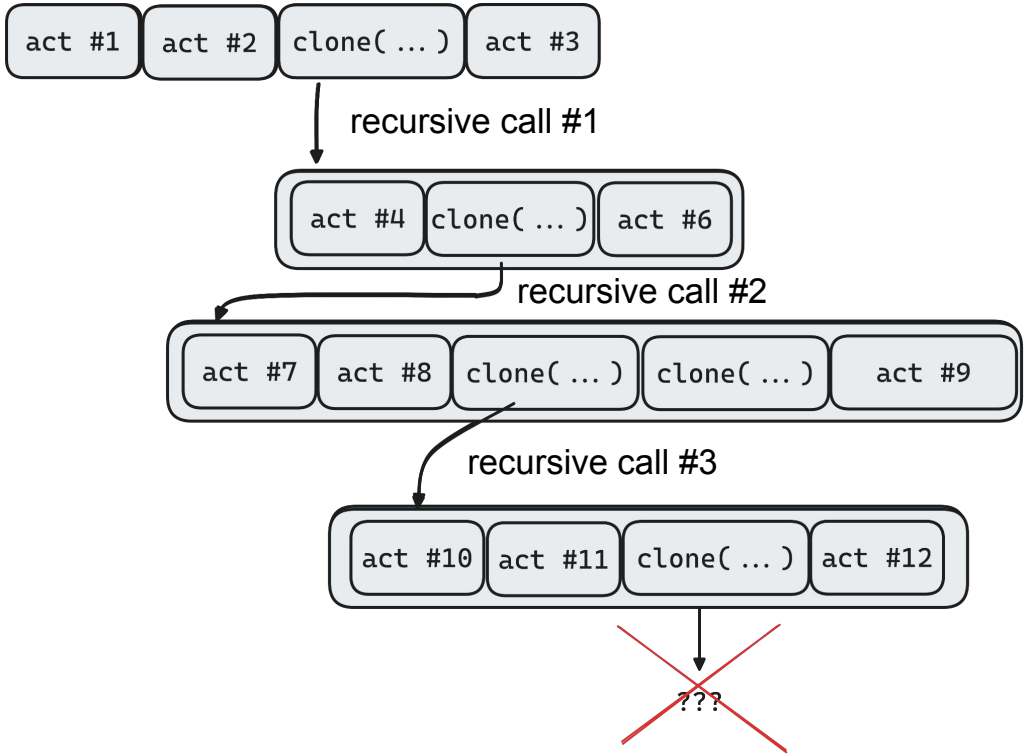



```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3
```

Deferred Actions FIFO:

1. act #13 | act #14 | act #15
2.
3.
4.
…
9.
10.

Netlink: Execution

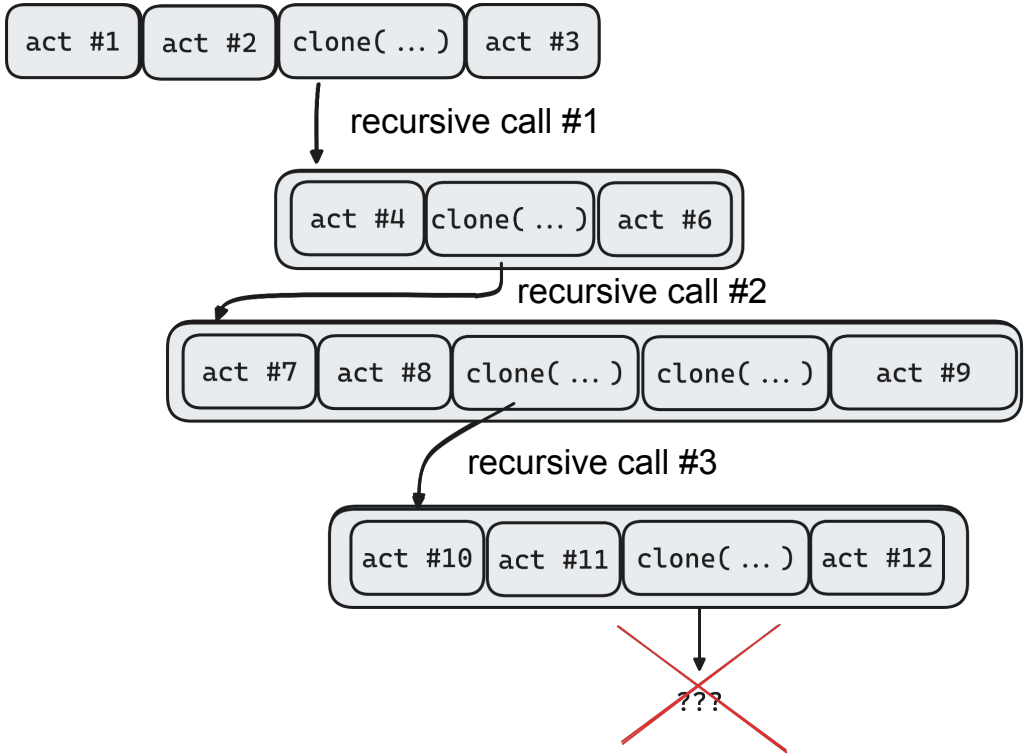



```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3
```

Deferred Actions FIFO:

1. act #13 | act #14 | act #15
2.
3.
4.
…
9.
10.

Netlink: Execution

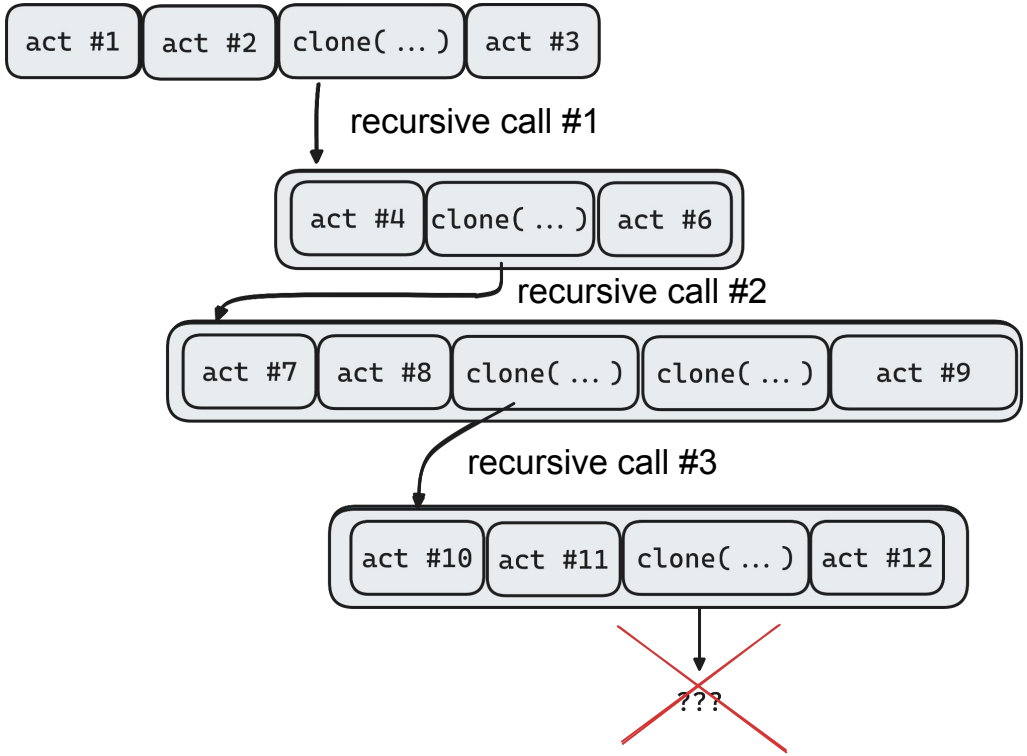



```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3
```

Deferred Actions FIFO:

1. act #13 | act #14 | act #15
2.
3.
4.
…
9.
10.

Netlink: Execution




```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3
```

Deferred Actions FIFO:

1. act #13 | act #14 | act #15
2.
3.
4.
…
9.
10.
11. ???

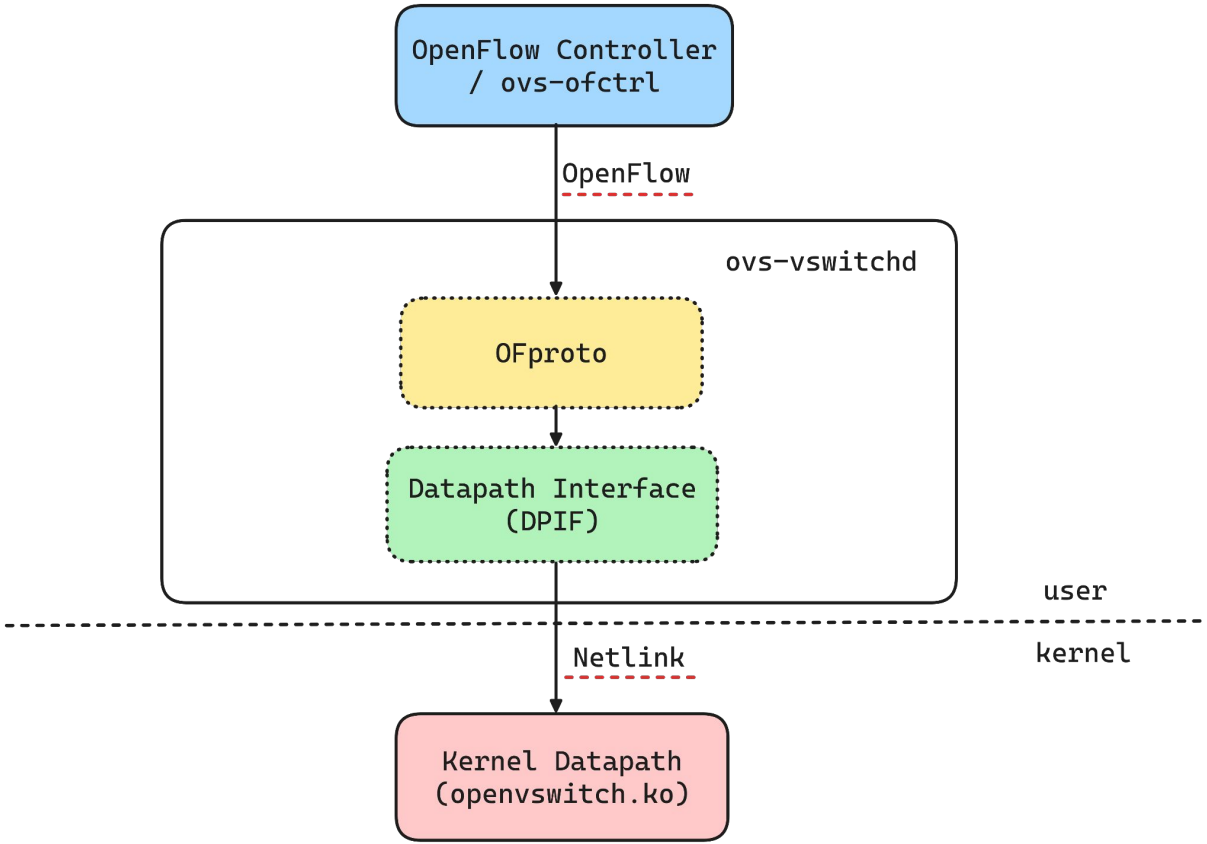
Netlink: Execution Warnings

```
OVS_RECURSION_LIMIT 5
OVS_DEFERRED_ACTION_THRESHOLD  3

DEFERRED_ACTION_FIFO_SIZE 10
```

```
kernel: recursion limit reached on datapath ...
kernel: deferred action limit reached, drop sample action
kernel: deferred action limit reached, drop recirc action (recirc_id=0x ... )
```

Protocols in OVS



OpenFlow

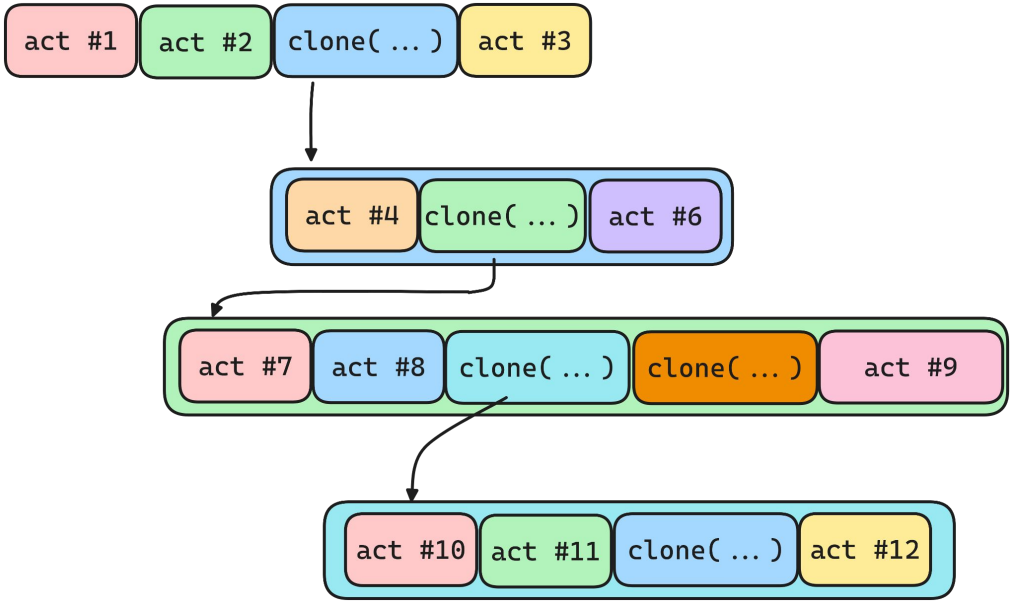
- In userspace:
  - Much more stack space when in the kernel: from 2 to 8 MB in many cases

OpenFlow

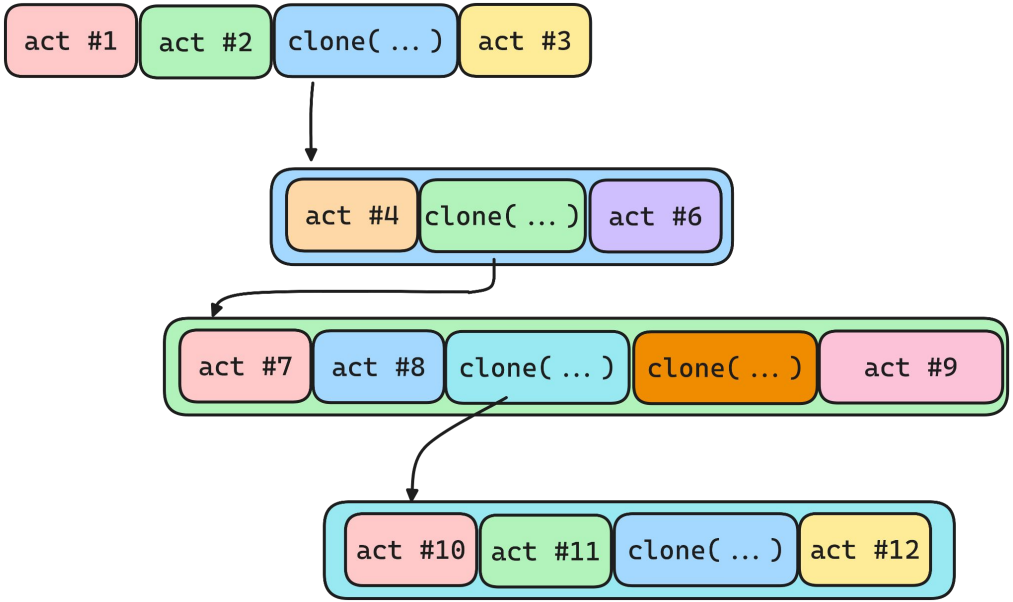
- In userspace:
  - Much more stack space when in the kernel: from 2 to 8 MB in many cases
- Much more complex processing…

```
table=0, in_port=p2 icmp action=controller(pause), resubmit(,1)
table=1, in_port=p2 icmp action=ct(table=2)
table=2, in_port=p2 icmp ct_state=+trk+new action=ct(commit, table=3)
table=3, in_port=p2 icmp action=p1
```

OpenFlow: Translation



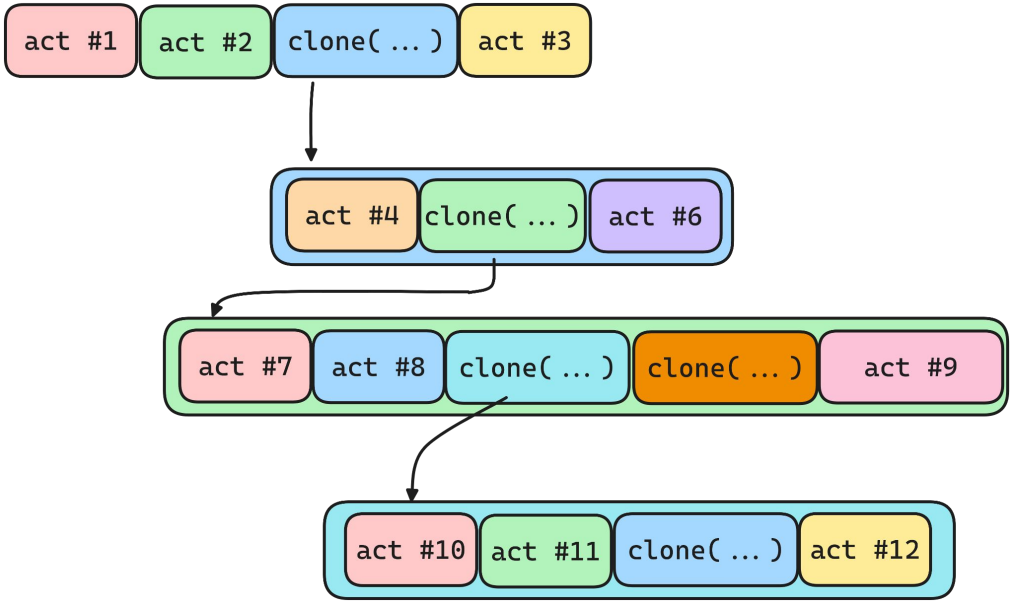
OpenFlow: Translation



That's a single OF rule.

1. It must fit into a single OF message, i.e. it's less than 64kB in size.

OpenFlow: Translation



That's a single OF rule.

1. It must fit into a single OF message, i.e. it's less than 64kB in size.
2. We assume that we can always translate a single OF rule.
3. But the nesting level is checked on initial parsing.

```
MAX_OFPACT_PARSE_DEPTH 100
```

OpenFlow: Translation – goto a different table

```
table=0, in_port=p2 icmp action=controller(pause), resubmit(,1)
table=1, in_port=p2 icmp action=ct(table=2)
table=2, in_port=p2 icmp ct_state=+trk+new action=ct(commit, table=3)
table=3, in_port=p2 icmp action=p1
```

OpenFlow: Translation – go to a different table

```
table=0, in_port=p2 icmp action=controller(pause), resubmit(,1)
table=1, in_port=p2 icmp action=ct(table=2)
table=2, in_port=p2 icmp ct_state=+trk+new action=ct(commit, table=3)
table=3, in_port=p2 icmp action=p1
```

Actions that direct flow translation into a different table:

- goto_table(N)
- resubmit(,N)
- ct(table=N)
- output:patch-port

OpenFlow: Translation – Limits

```
MAX_DEPTH 64
MAX_RESUBMITS 4096
```

OpenFlow: Translation – Limits

What is counted towards depth?

```
MAX_DEPTH 64
MAX_RESUBMITS 4096
```

- Only counted moves to a different table, when the table ID is lower than the current one.

  8 MB / 255 tables / 64 depth ~ 512 B per OF rule

OpenFlow: Translation – Limits

What is counted towards depth?

```
MAX_DEPTH 64
MAX_RESUBMITS 4096
```

- Only counted moves to a different table, when the table ID is lower than the current one.

  8 MB / 255 tables / 64 depth ~ 512 B per OF rule

What is counted towards resubmits?

- Any jump to a different table (most of them).

  8 MB / 4096 resubmits ~ 2 kB per OF rule
  2 MB / 4096 resubmits ~ 512 B per OF rule

OpenFlow: Translation – Errors

```
MAX_DEPTH 64
MAX_RESUBMITS 4096
```

```
over max translation depth 64
over 4096 resubmit actions
```

OpenFlow: Translation

512 B of stack per OF rule…

OpenFlow: Translation

512 B of stack per OF rule…

```
sizeof(struct flow) = 672 B
sizeof(struct match) = 3400 B
```

- Need to reduce stack usage by individual functions. *
- Need a generic solution that depends on the actual stack usage.

*4829506b2a21 ("ofproto-dpif-xlate: Reduce stack usage in recursive xlate functions.") [Mike Pattrick]

OpenFlow: Translation – Stack usage tracking

- __builtin_frame_address(0)
- pthread_attr_getstacksize()

OpenFlow: Translation – Stack usage tracking

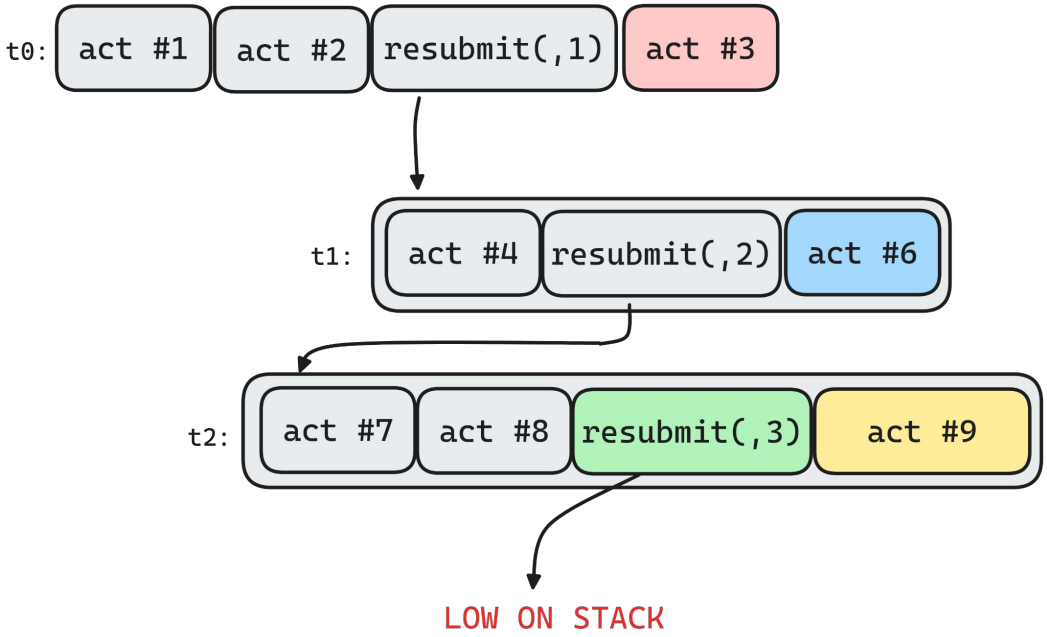
- __builtin_frame_address(0)
- pthread_attr_getstacksize()

Trigger recirculation when the stack frame address is close to the base frame address + stack size:

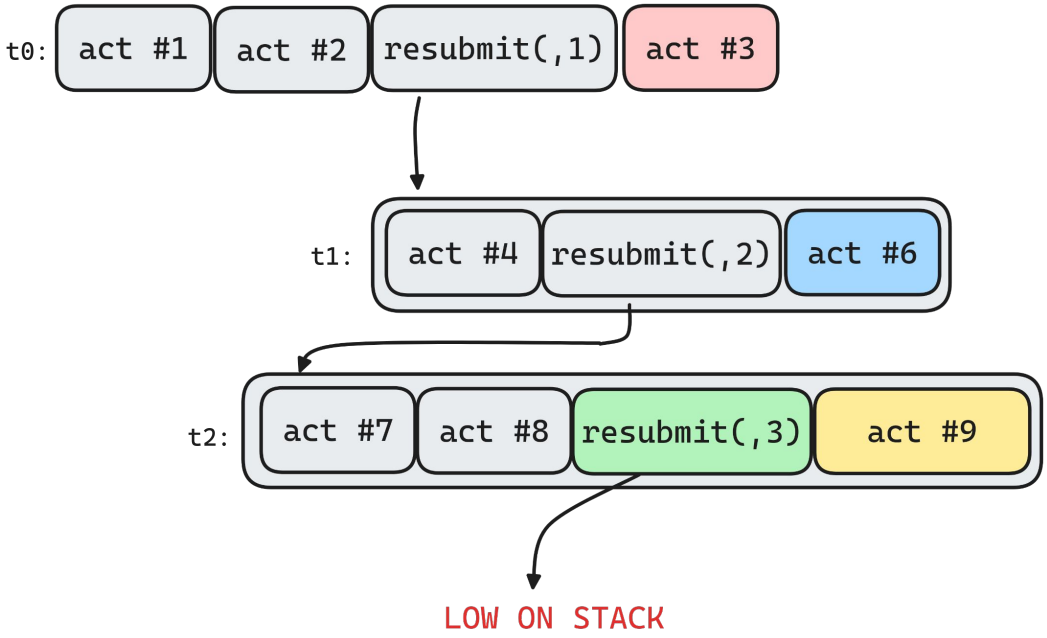
- [ovs-dev,RFC] ofproto-dpif-xlate: Recirculate on stack exhaustion.

  https://patchwork.ozlabs.org/project/openvswitch/patch/20240223012704.2793017-1-i.maximets@ovn.org/

OpenFlow: Translation – Stack usage tracking



OpenFlow: Translation – Stack usage tracking

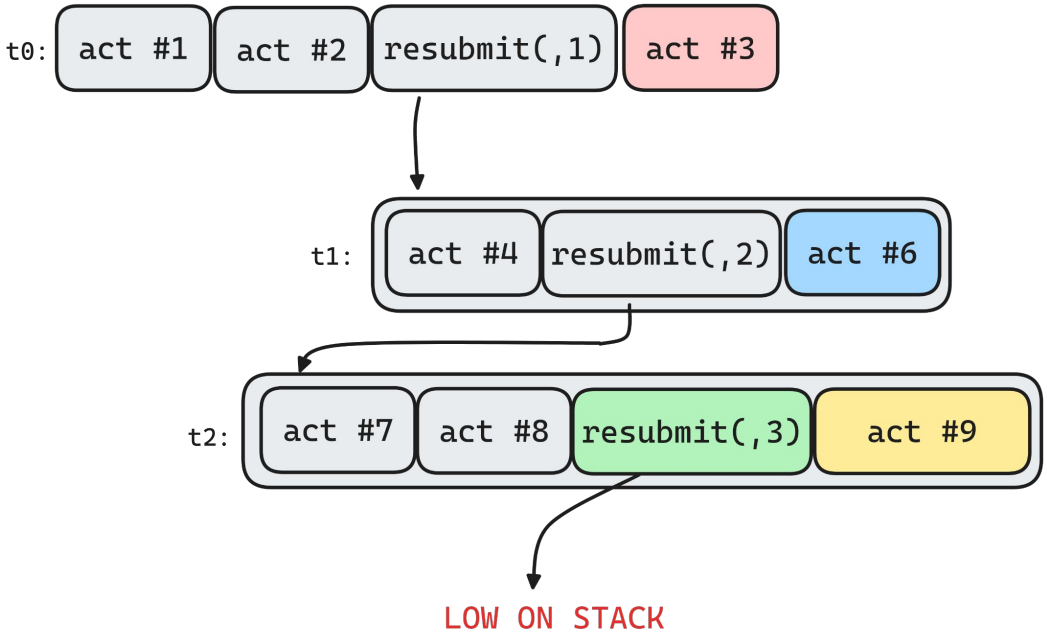


Frozen State

- Frozen Actions:

- Flow key
- Other metadata
- ID: 0xabcdef

OpenFlow: Translation – Stack usage tracking

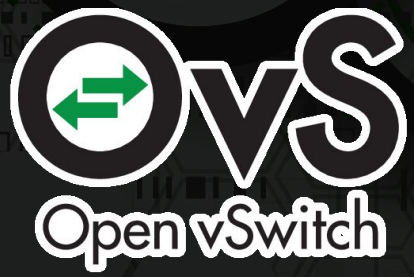


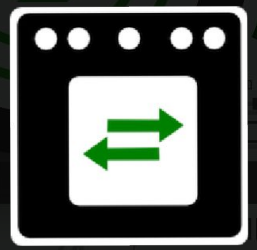
Frozen State

- Frozen Actions:

- Flow key
- Other metadata
- ID: 0xabcdef

```
actions: ... , recirc(0×abcdef)
```





Thanks!

Email: i.maximets@ovn.org