Deparsing Proposal

Leo Alterman // Barefoot Networks

August 31st, 2015

Problem

Deparsing

- Current deparsing definition:

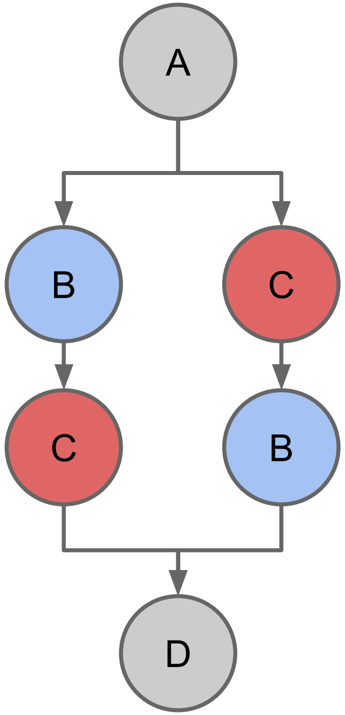
  P4 takes the approach that any format which should be generated on egress should be represented by the parser used on ingress. Thus, the parse graph represented in the P4 program is used to determine the algorithm used to produce the serialized packet from the Parsed Representation. Note the following considerations:

  - Only headers which are valid are serialized.
  - If the parse graph is acyclic, then a topological ordering (that is, a linear order that respects the parse graph’s ordering) can be generated and used to determine the order by which headers should be serialized.
  - In general, cycles occur in the parse graph when parsing header stacks or a set of optional headers. These may be treated as a single node in the parse graph and serialized as a group.

Deparsing

- Problems with this:
  - Parse graphs without a topological ordering
    - Shows up for TLV headers
  - Inserting headers before ethernet
    - Shows up for fabric headers
  - Checksum updates
    - Some targets may perform checksum calculations only after general Match+Action processing

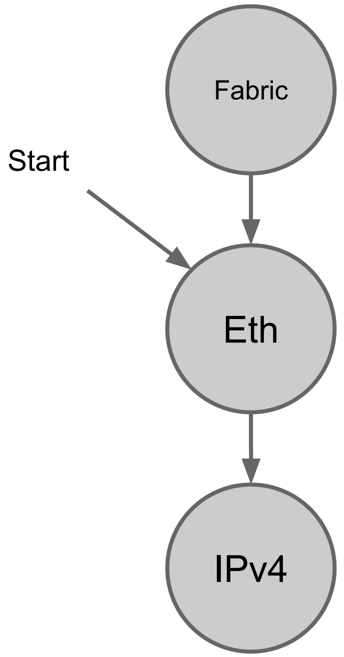
Non-topological parse graph


$$\begin{tikzcd}
 & A \arrow[dl] \arrow[dr] & \\
B \arrow[d] & & C \arrow[d] \\
C \arrow[dr] & & B \arrow[dl] \\
 & D &
\end{tikzcd}$$

- This is the simplest example of a parse graph without a topological header ordering

- What should the deparsing behavior be?
    - Preserve original ordering?
        - Some targets might not be able to
        - What if we're adding these headers from scratch?
    - Specify some canonical ordering?
        - Might reorder headers - for things like IPv4 options, this is probably okay

Inserting headers before Ethernet



- Or generally: how do you insert *new* headers on top of the ones you already parse?

- Can currently hack in "ghost" states that are in the parse graph but have transitions which can never be taken

- Better solutions?
    - Specify a canonical header ordering wtih 'Fabric' at top?
    - Specify a header instance always directly preceeds/follows another header instance?

Checksum updates

- Or generally: how do you express simple packet modifications that happen after the match action pipeline?

- Both v1 and current proposal sidestep this issue with slightly bizarre 'checksum update' syntax

- Deparsing is a nice place to put checksum updates
  - Don't have to worry about modifying checksums incrementally in middle of program (which could affect code modularity)
  - Naturally expressed as reassigning a field's value before emitting it

Solution

(or at least, a beginning of a solution)

Explicit Deparsing

- Hard to tell at this point what the right abstraction for deparsing is
  - Needs to make sense at the program level
  - Needs to work on a variety of targets

- Proposal: Start by explicitly exposing 'packet interface' to get and set serialized data, then allow target architectures to figure out the right way to use it

Packet Blackbox

- Packet blackbox:
    - extract() method, replaces built-in parser extract() function
    - emit() method, added for deparsing
        - takes a header instance - outputs to header wire if it's valid, does nothing if not valid

- Packet blackbox instance is passed into parser and deparser whiteboxes as an argument (user never creates their own packet objects)

- Target architecture may or may not expose a deparser whitebox, may make it optional, etc.

Packet Blackbox Parser Example

```
whitebox_type parser_module (blackbox packet_t p, struct my_headers_t h) {

      parser start {
            p.extract(h.ethernet);
            return select (h.ethernet.ethertype) {
                  ETH_IPV4: parse_ipv4;
                  ETH_IPV6: parse_ipv6;
            }
      }

      parser ipv4 {
            p.extract(h.ipv4);
            return select (h.ipv4.proto) {
                  // …
            }
      }

      // …
}
```

Packet Blackbox Deparser Example

```
whitebox_type deparser_module (blackbox packet_t p, struct my_headers_t h) {

     // Architecture disallows anything inside this control func other
     // than packet 'emit' calls and checksum rewrites

     blackbox checksum ipv4_chksum {
          fields {
               // list every h.ipv4 field here other than 'chksum'
          }
          algorithm : csum16;
     }

     control deparse {
          p.emit(h.fabric_header);
          p.emit(h.ethernet);
          ipv4_chksum.get_value(h.ipv4.chksum);
          p.emit(h.ipv4);
          p.emit(h.ipv6);
          // …
     }
}
```