A PNA proposal to standardize checksum externs

Proposal to standardize checksum externs

1. DPU needs to perform checksum verification of incoming packet and compute checksum for outgoing packet
2. L3, L4 Checksum verification
   1. Verify more than one layer of L3 and L4 checksums.
   2. Provide checksum verification result.
   3. Skip verification of zero checksum value. (For certain packet encapsulation, it's fine to have zero UDP checksum)
3. L3, L4 Checksum computation
   1. Compute more than one layer of L3 and L4 checksums
   2. Include computed checksum of inner packet layer/s into outer layer's packet checksum.
4. Other checksum computation
   1. Compute complete checksum for Linux kernel to use (Rx Packet)

Proposal to standardize checksum externs

1. Unified checksum extern class and methods for checksum verification and computation
2. Depending upon DPU architecture checksum objects and methods are used in appropriate stages in pipeline. Define methods to trigger verification, obtain verified results and to trigger checksum computation.
3. Generally, checksum fields of a packet is verified inside parser. The extern objects and methods could be used in Parser block.
4. Pipeline may trigger checksum computation. There may be a need for the extern objects and methods to be used inside pipeline or action routines

Extern class and methods

```
extern Checksum16 {
    Checksum16();
    Checksum16(bool csum_zero_skip_verify);
    void validate<F>(in F checksum_destination);
    bit<16> get();
    bit<1> validation_failed();
    void update_len<O, L>(in O offset, in L length);
    void include_checksum_result<H>(in H checksum_header);
    void update_pseudo_header_offset<H, O>(in H hdr, in O offset);
    void update_pseudo_header_fields<P, L>(in P pseudo_hdr, in L fields);
    void update_pseudo_header_constant(in bit<8> next_hdr);
    void enable_update();
    void enable_update<H>(in H hdr);
    void disable_update();
    void disable_update<H>(in H hdr);
    void compute_complete_checksum<H, L>(in H hdr, in L length);
    void compute_complete_checksum_after<H, L>( in H hdr, in L length);
}
```

- Extern class and methods. Next few slides will describe each of these methods in detail.

Extern class and constructors

```
extern Checksum16 {
    Checksum16();
    Checksum16(bool csum_zero_skip_verify);
}
```

- Default constructor and a variant constructor with an ability to skip verification of packet checksum with zero value
- Computes 16 bits checksum value.

Extern methods to validate, compute, check verification results

```
extern Checksum16 {
      void validate<F>(in F checksum_destination);
      bit<16> get();
      bit<1> validation_failed();
}
```

- The validate() method allows to verify incoming packets checksum value.
  - checksum_destination : Packet header field where checksum value is present in a packet. This parameter is a header field. (Example: header.ipv4.checksum)
- The get() method computes checksum value. The method returns 16 bits two's complement over the packet data. Data over which checksum is computed is provided using update_len() method [Details in next slide].
- The validation_failed() method allows pipeline to check checksum verification result.
  When validation fails, the method returns 1.

Extern methods to setup checksum bytes

```
extern Checksum16 {
      void update_len<O, L>(in O offset, in L length);
      }
```

- The update_len() method uses offset into packet and length in bytes. Depending on get() method or verify() method, checksum is computed or verified over the length bytes starting from the offset.

Extern method for handling multi layer packets.

```
extern Checksum16 {
    void include_checksum_result<H>(in H checksum_header);
}
```

- This method allows to include checksum computation results of inner headers into checksum computation being performed on outer headers (Example : When outer UDP checksum and inner UDP checksum are being computed, include inner UDP checksum result into outer UDP checksum computation).

Extern methods related to L4 checksums with pseudo header fields.

```
extern Checksum16 {
      void update_pseudo_header_offset<H, O>(
          in H hdr, in O offset);

      void update_pseudo_header_fields<P, L>(
          in P pseudo_hdr, in L fields);

      void update_pseudo_header_constant(
          in bit<8> next_hdr);
}
```

- The update_pseudo_header_offset() method uses offset into packet where Layer3 header starts. Checksum verification requires to collect a set of pseudo header fields. By providing both start of pseudo header and a list of pseudo header fields (method to specify this list of fields in the next slide), layer 4 checksum verification can be achieved.

- The method update_pseudo_header_fields() provides an ability to include pseudo header fields in L4 checksum.
    - pseudo_hdr : A well known layer 3 packet header instance like ipv4/ipv6/ etc.
    - fields : A list of layer 3 packet header fields, constants, user metadata fields or parser local variable to be included in layer 4 checksum computation or verification.
- The method udpate_pseudo_header_constant() allows to include constant value into checksum verification or computation. This is useful when certain pseudo header fields are not known to be present at a well known offsets. Example next_header field in ipv6 header may be at varying offset if v6 options are present.

Extern methods to enable computation

```
extern Checksum16 {
      void enable_update();
      void enable_update<H>(in H hdr);
}
```

- The enable_update() method allows to trigger checksum computation. An example usage of this method is in p4-action

- The enable_update<H>(in H hdr) method allows to trigger checksum computation. This method also uses header as argument. This argument helps to collect pseudo header fields provided in update_pseudo_header_fields<P, L>() method from the header "hdr" . An example usage of this method is in p4-action

Extern methods to disable computation

```
extern Checksum16 {
      void disable_update();
      void disable_update<H>(in H hdr);
}
```

- The disable_update() method allows to disable checksum computation. An example usage of this method is in p4-action. In error / drop scenario, pipeline can disable checksum computation using this method.
- The disable_update<H>(in H hdr) method allows to disable checksum computation. An example usage of this method is in p4-action. In error / drop scenario, pipeline can disable checksum computation using this method.

Other extern methods to compute checksum

```
extern Checksum16 {
   void compute_complete_checksum<H, L>(
            in H hdr, in L length);

   void compute_complete_checksum_after<H, L>(
            in H hdr, in L length);
}
```

- The method compute_complete_checksum<H,L>(in H hdr, in L length) computes checksum starting from the header "hdr" over "length" bytes. The method can be invoked more than once using outer, inner or inner-inner headers. Depending upon the presence or outer, inner or inner-inner header in the outgoing packet, enable_update<H> method will choose the starting point from where checksum will be computed.
  - Suppose output going packet has two ethernet headers , ethernet_outer, ethernet_inner, if enable_update(header.ethernet_outer) is invoked in pipeline, then checksum from the start of ethernet_outer will be computed. If both enable_udpate(header.ethernet_outer) and enable_update(header.ethernet_inner) are invoked depending upon most recent method invoked in the pipeline, corresponding header will be used as start point for computation.
- The method compute_complete_checksum_after<H,L>(in H hdr, in L length) is similar to compute_complete_checksum() except that starting point of checksum computation is from the header that follows the header "hdr" in the outgoing packet.

Example: Verify ipv4.checksum

```
 Checksum16()          ipv4HdrCsum;
parser  p(packet_in packet, out headers hdr,.. ) {
  bit<16>                      l3_hdr_offset;
  :
  state parse_ipv4 {
    l3_hdr_offset = packet.state_byte_offset();
    packet.extract(hdr.ipv4);
    transition parse_ipv4_checksum;
  }
  state parse_ipv4_checksum {
    bit<16>     ip_header_len;
    ip_header_len = ((bit<16>) hdr.ipv4.ihl << 2);
    ipv4HdrCsum.update_len(l3_hdr_offset, ip_header_len);
    ipv4HdrCsum.validate(hdr.ipv4.checksum);
 }
  :
}
```

- Code snippet to show how checksum extern object and methods will be used to verify ip header checksum.
- The result of the checksum verification can be obtained in match action pipeline using ipv4HdrCsum.validation_failed()

Example: Verify L4 checksum (udp or tcp or icmp)

```
 Checksum16()          udpCsum;
parser  p(packet_in packet, out headers hdr,.. ) {
  bit<16>                          l3_hdr_offset;
  :
  state parse_ipv4 {
    l3_hdr_offset  = packet.state_byte_offset();
    packet.extract(hdr.ipv4);
    transition parse_ipv4_checksum;
  }
  state parse_ipv4_checksum {
    bit<16>     ip_header_len;
    ip_total_len   = hdr.ipv4.totalLen;
    ip_header_len = ((bit<16>) hdr.ipv4.ihl  << 2)
    l4_len         = ip_total_len - ip_header_len;
    udpCsum.update_pseudo_header_offset(hdr.ipv4,  l3_hdr_offset);
    udpCsum.update_pseudo_header_fields(hdr.ipv4,
                     {hdr.ipv4.srcAddr,  hdr.ipv4.dstAddr,  l4_len});
    udpCsum.update_pseudo_header_constant(IP_PROTO_UDP);
 }
  :
```

```
 :
  state parse_udp {
    packet.extract(hdr.udp);
    l4_hdr_offset  = packet.state_byte_offset();
    udpCsum.update_len(l4_hdr_offset,  l4_len)
    udpCsum.validate(hdr.udp.checksum);
  }
 }
```

- Code snippet to show how checksum extern object and methods will be used to verify udp checksum.
- The result of the checksum verification can be obtained in match action pipeline using udpCsum.validation_failed()
- The example uses Ipv4 as L3 header. Pseudo header fields for ipv6 can added as well using these methods. At packet processing time, depending on the presence of ipv4 or ipv6 header, checksum block will use appropriate pseudo header fields.

Example: Compute ipv4.checksum

```
Checksum16()          ipv4HdrCsum;
 control deparser (packet_out packet, inout headers hdr, control_metadata
     metadata, .. ) {
   apply {
     packet.emit(hdr.ipv4);
     ipv4HdrCsum.update_len(hdr.ipv4,  metadata.csum.ip_hdr_len);
     hdr.ipv4.checksum  = ipv4HdrCsum.get();
   }
}

control match-action ( inout headers hdr, control_metadata  metadata, .. ) {
   action  foo() {
     ipv4HdrCsum.enable_update();
   }
 }
```

- Code snippet to show how checksum extern object and methods will be used to compute ip header checksum.
- The result of the checksum computation can be obtained and set into ipv4 header checksum field using get() method

Example: Compute tcp.checksum

```
Checksum16()          tcpCsum;
 control deparser (packet_out packet, inout headers hdr, control_metadata
     metadata, .. ) {
   apply {
     packet.emit(hdr.tcp);
     tcpCsum.update_pseudo_header_fields(hdr.ipv6,  {hdr.ipv6.srcAddr,
     hdr.ipv6.dstAddr,  metadata.csum.tcp_len});
     tcpCsum.update_pseudo_header_constant(IP_PROTO_TCP);
     tcpCsum.update_len(hdr.tcp,  metadata.csum.tcp_len);
     hdr.tcp.checksum  = tcpCsum.get();
   }
 }

control match-action ( inout headers hdr, control_metadata  metadata, .. ) {
   action  foo() {
     tcpCsum.enable_update();
   }
 }
```

- Code snippet to show how checksum extern object and methods will be used to compute tcp checksum.
- The result of the checksum computation can be obtained and set into tcp header checksum field using get() method.
- Since ipv6 next header field can be at non fixed offset from the start of the header (depends on ipv6 options), for tcp checksum with v6 L3 header, TCP protocol constant value is added as pseudo header field.