

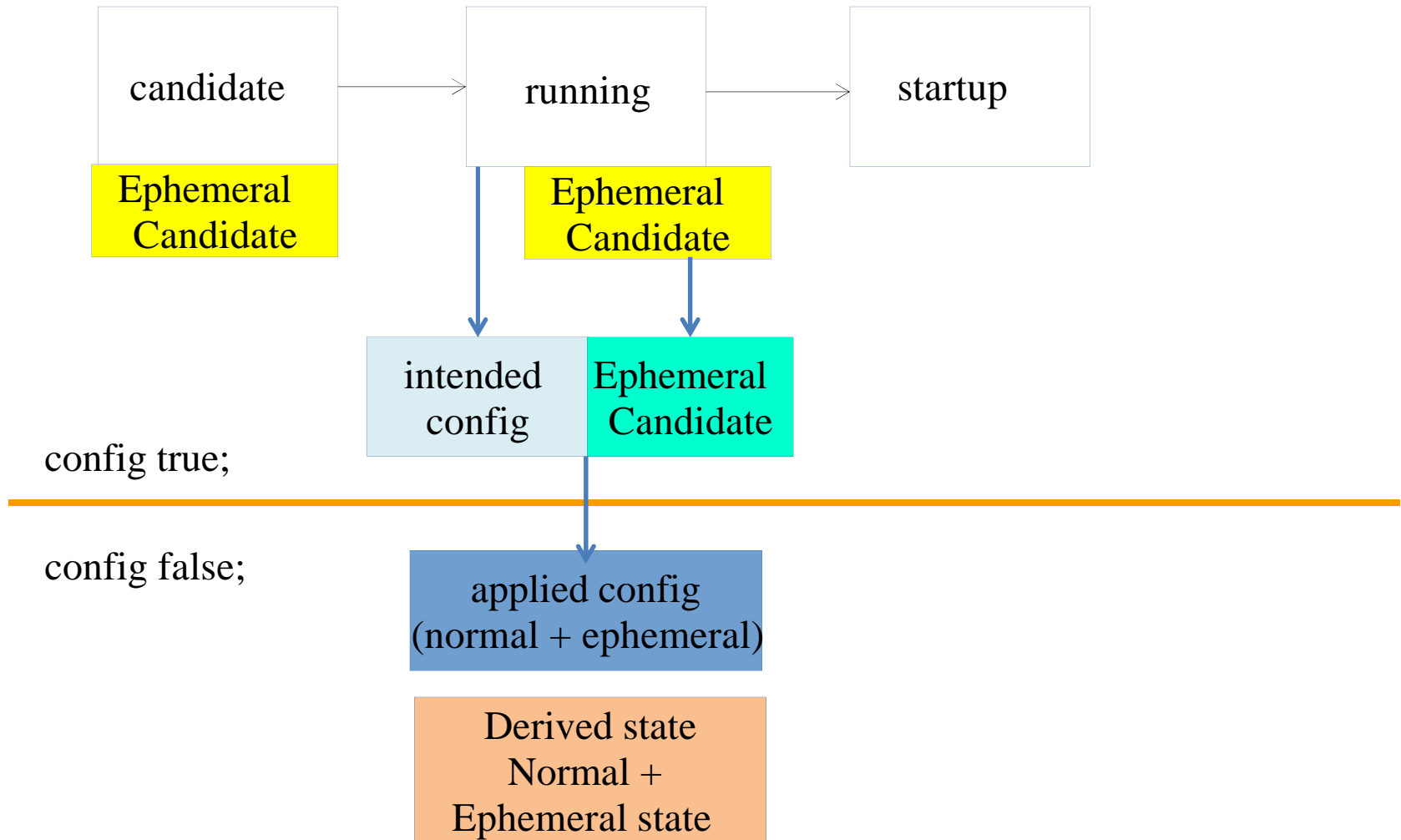
I2RS RIB Route Example

Sue Hares

I2RS RIB Example

- First Case
 - 128.2/16 with nexthop 1 – added by netconf config
 - 128.2/16 with nexthop 2 – added by I2RS RIB
 - DDOS attack causes you to overwrite NETCONF config with I2RS RIB route

Ephemeral Additions



Route

```
module i2rs-rib {  
  ...  
  container routing-instance {  
    ...  
    list rib-list {  
      ...  
      list route-list {  
        key "route-index";  
        uses route;  
      }  
    }  
  }  
}
```

operational
data

Extensions

```
grouping route {  
  description  
    "The common attribute used for all routes;"  
  uses route-prefix;  
  container nexthop {  
    uses nexthop;  
  }  
  container route-statistics {  
    leaf route-state {  
      type route-state-def;  
      config false; /* operational state */  
    }  
    leaf route-installed state {  
      type route-installed-state def;  
      config false;  
    }  
    leaf route-reason {  
      type route-reason-def;  
      config false;  
    }  
  }  
  container router-attributes {  
    uses router-attributes;  
  }  
  container route-vendor-attributes  
    uses route-vendor attributes;  
}
```

Route

Index for route direct reference without prefix match; Main key.

Type: ipv4, ipv6, mpls, mac, interface

Type: v4 prefix match

Index for nexthop direct index without match

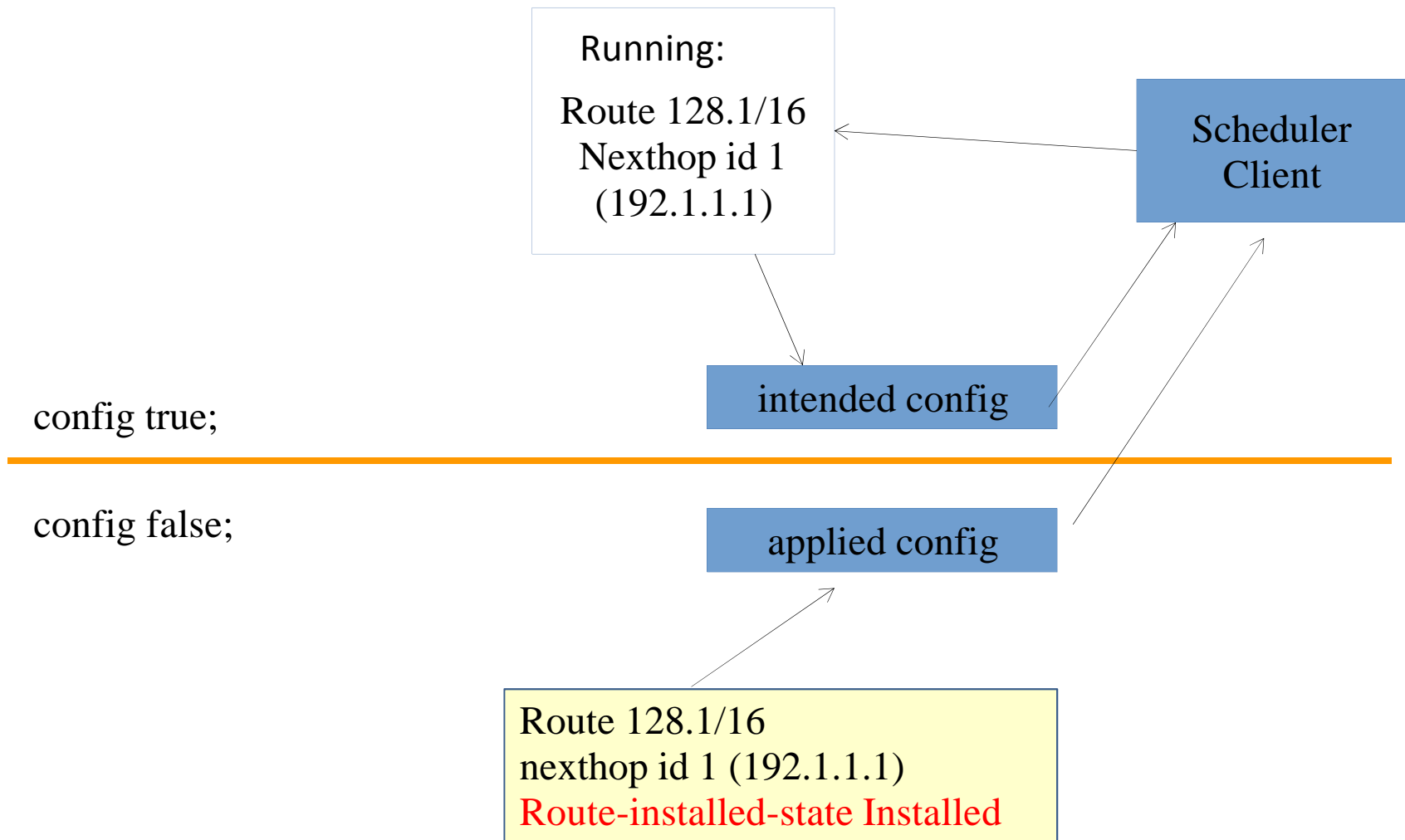
IPv4 prefix

```
module i2rs-rib { ....
  container routing-instance { ...
    list rib-list { ....
      list route-list {
        key "route-index";
        leaf route-index {
          type uint64;
          mandatory true;
        }
        leaf route-type {
          type route-type-def;
          mandatory true;
        }
        Container match {
          choice rib-route-type { ....
            leaf destination-ip-v4-prefix {
              type inet:ipv4-prefix;
              mandatory true;
            }
          }
        }
        leaf nexthop-id {
          type uint32;
          mandatory true;
        }
        leaf next-hopo-ipv4-address {
          type inet:ipv4-prefix;
          mandatory true;
        }
      }
    }
  }
}
```

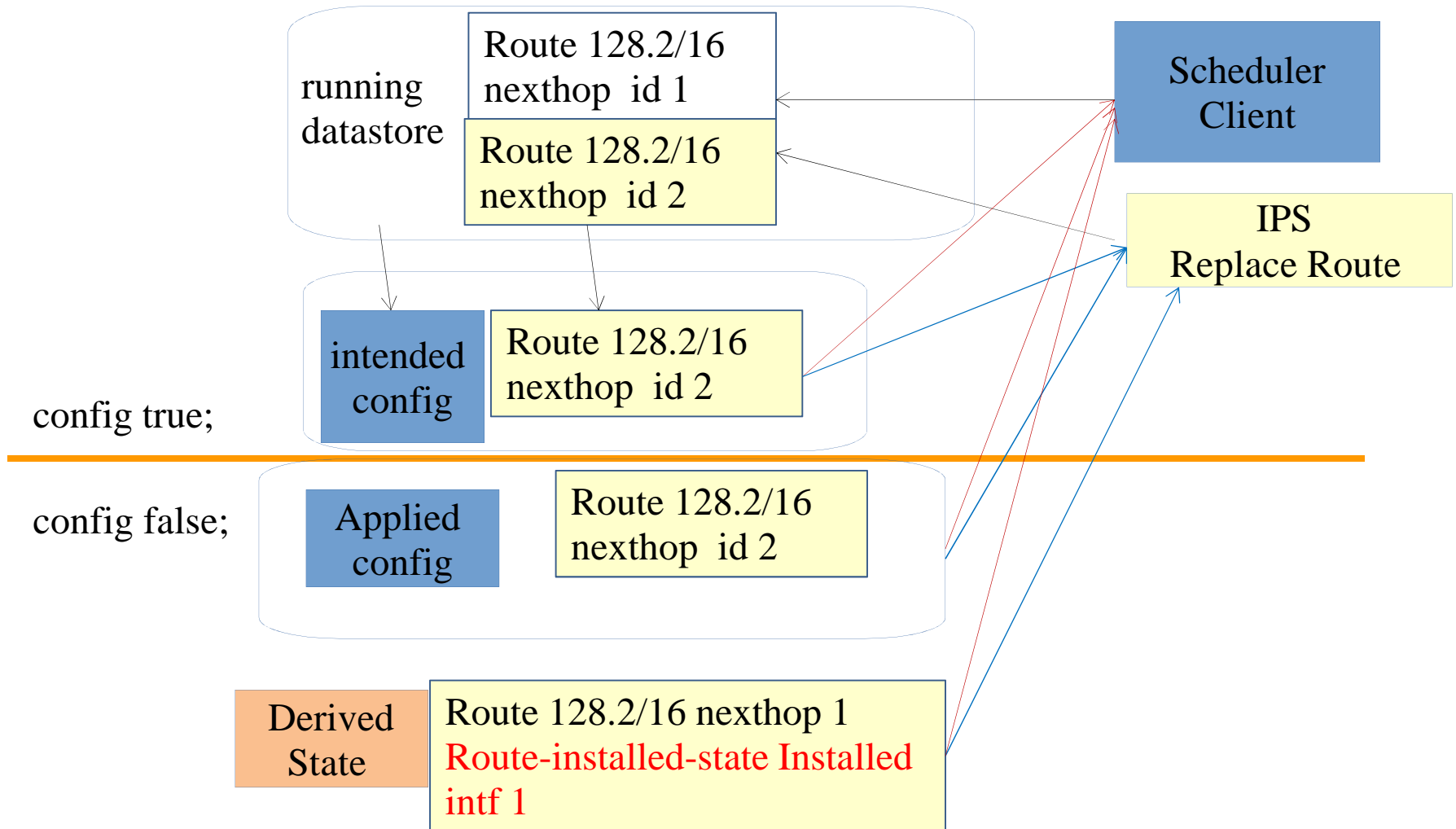
```
container route-statistics {
  leaf route-installed state {
    type route-installed-state def;
    config false;
  }
}
```

Defined as:
Installed, uninstalled

Thermostat Model RIB Equivalent



Route + Ephemeral Route



RESTCONF Example

RESTCONF Running Datastore Edit

```
PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2/next-hop  
{ "next-hop":1 }
```

RESTCONF Ephemeral Datastore Edit of config=true

```
PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2/next-hop?datastore=ephemeral  
{ "next-hop":2 }
```

RESTCONF Ephemeral Datastore Read of config=false

```
GET /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.1/next-hop=2/route-installed-state/datastore=ephemeral  
{ "route-installed-state" }
```


RESTCONF Example

RESTCONF Ephemeral Datastore Edit of config=false

```
PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.1/next-  
hop=2/route-installed-state/context=ephemeral  
{ "route-installed-state": Installed }
```

Route

Index for route direct reference without prefix match; Main key.

Type: ipv4, ipv6, mpls, mac, interface

Type: v4 prefix match

Index for nexthop direct index without match

IPv4 prefix

```
module i2rs-rib { ....
  container routing-instance { ...
    list rib-list { ....
      list route-list {
        key "route-index";
        leaf route-index {
          type uint64;
          mandatory true;
        }
        leaf route-type {
          type route-type-def;
          mandatory true;
        }
        Container match {
          choice rib-route-type { ....
            leaf destination-ip-v4-prefix {
              type inet:ipv4-prefix;
              mandatory true;
            }
          }
        }
        leaf nexthop-id {
          type uint32;
          mandatory true;
        }
        leaf next-hopo-ipv4-address {
          type inet:ipv4-prefix;
          mandatory true
        }
      }
    }
  }
}
```

Add
Ephemeral true;

```
container route-statistics {
  leaf route-installed state {
    type route-installed-state def;
    config false;
  }
}
```

Defined as:
Installed, uninstalled

RESTCONF Example

RESTCONF Running Datastore Edit

```
PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2/next-hop  
{ "next-hop":1 }
```

RESTCONF Ephemeral Datastore Edit of config=true

```
PUT /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2.1/next-  
hop?datastore=ephemeral  
{ "next-hop":2 }
```

RESTCONF Ephemeral Datastore Edit of config=false

```
GET /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2.1/next-  
hop=2/route-installed-state/datastore=ephemeral  
{ "route-installed-state": Installed }
```

RESTCONF Example

RESTCONF Ephemeral Datastore Edit of config=false

```
Put /restconf/data/i2rs-rib/instance=1/rib=IPv4/route=128.2.1/next-  
hop=2/route-installed-state/datastore=ephemeral  
{ "route-installed-state": Installed }
```

Should we do this or an IPC

RIB-DM Design issues

- Next-hop protection
- Overlapping routes

Nexthop Protection

nexthop-id - integer.

<nexthop> ::= <NEXTHOP_PROTECTION>

<1> <interface-primary>

<2> <interface-backup>

Protection-id 1: preference=10, nexthop-id=1

Protection-id 2: preference = 2, nexthop-id=1

Protection-id 3: preference=1, nexthop-id = 1

Protection-id 4: preference =1, nexthop-id=2

Batch and overlapping installs

- Suggest
 - Huawei uses an RPC
 - ODL uses large PUT
 - RESTCONF supports PATCH
- Large batch with some overlap
 - Flag for individual route installs makes sense for some implemented
 - Flag for all-or-nothing makes sense for others

Backup on creating the shorten route

**FROM I2RS YANG MODULE TO
SHORT ROUTE**


```

module i2rs-rib {
  ...
  container routing-instance {
    ...
    list rib-list {
      ...
      list route-list {
        key "route-index";
        uses route;
      }
    }
  }
}

```

```

grouping route {
  description
    "The common attribute
    used for all routes;"
  uses route-prefix;
  container nexthop {
    uses nexthop;
  }
  ....
}
grouping route-prefix {
  description "common
  attributes use for all routes";
  leaf route-index {
    type uint64;
    mandatory true;
  }
  leaf route-type {
    type route-type-def;
    mandatory true;
  }
  container match {
    choice rib-route-type {
      ... ipv4
      ... ipv6
      ... mpls
      ... mac
    }
  }
}

```

```

grouping nexthop {
  leaf nexthop-id {
    mandatory true;
    type uint32;
  }
  choice next-hop-type {
    case next-hop base {
      list nexthop-chain {
        key "nexthop-chain-id";
        uses nexthop-chain-member;
      }
    }
  }
}

```

```

case ipv4 {
  description
    "match on destination IP
    address in header";
  container ipv4 {
    leaf ipv4-route-type {
      type ip-route-type def;
      mandatory true;
    }
    choice ip-route-type {
      case destination-ipv4-address {
        leaf destination-ipv4-prefix {
          type inet:ipv4-prefix;
          mandatory true;
        }
      }
      case destination-source-ipv4-address
        ....
    }
  }
}

```

Route info

```
module i2rs-rib {
  ...
  container routing-instance {
    ...
    list rib-list {
      ...
      list route-list {
        key "route-index";
        leaf route-index {
          type uint64;
          mandatory true;
        }
        leaf route-type {
          type route-type-def;
          mandatory true;
        }
        leaf destination-ip-v4-prefix {
          type inet:ipv4-prefix;
          mandatory true;
        }
        left nexthop-id {
          type uint32;
          mandatory true;
        }
        leaf next-hop-ipv4-address {
          type inet:ipv4-address
          mandatory true
        }
      }
    }
  }
}
```

IPv4
Route

```
grouping route {
  description
  "The common attribute
  used for all routes;"
  uses route-prefix;
  container nexthop {
    uses nexthop;
  }
  ....
}
grouping route-prefix {
  description "common
  attributes use for all routes";
  leaf route-index {
    type uint64;
    mandatory true;
  }
  leaf route-type {
    type route-type-def;
    mandatory true;
  }
  container match {
    choice route-type {
      ... ipv4
      ... ipv6
      ... mpls
      ... mac
    }
  }
}
```

```
grouping nexthop {
  leaf nexthop-id {
    mandatory true;
    type uint32;
  }
  choice next-hop-type {
    case next-hop base {
      list nexthop-chain {
        key "nexthop-chain-id";
        uses nexthop-chain-member;
      }
    }
  }
}
```

```
case ipv4 {
  description
  "match on destination IP
  address in header";
  container ipv4 {
    leaf ipv4-route-type {
      type ip-route-type def;
      mandatory true;
    }
    choice ip-route-type {
      case destination-ipv4-address {
        leaf destination-ipv4-prefix {
          type inet:ipv4-prefix
          mandatory true;
        }
      }
      case destination-source-ipv4-
      address
      ....
    }
  }
}
```