Efficient P4-Based BIER Implementation on Tofino

Steffen Lindner, Daniel Merling, and Michael Menth

http://kn.inf.uni-tuebingen.de
► Previous Work

► Problem Statement

► Implementation of Efficient BIER in P4

► Optimization
IETF 108

- First implementation of BIER & BIER-FRR in P4 @ Intel™ Tofino
- Iterative processing
  - In each iteration, one next-hop is served
Caveat

- Recirculation requires capacity
- 100 Gbit/s multicast traffic with 5 next-hops results in **400 Gbit/s recirculation traffic**
- **Solution**: Add dedicated recirculation ports to increase recirculation capacity
Problem Statement

► Goal: Reduce recirculations to improve efficiency

► Idea
  ▪ Determine all next-hops in (possibly) one shot
  ▪ Use a (static) internal multicast group to replicate packets to next-hops

► Challenges
  ▪ BIER bitstrings with 256+ bits difficult to map to the set of packet’s next-hops in a single table lookup (without specialized hardware)
  ▪ Limited number of configurable static multicast groups

How can we do this more efficiently?
Efficient Implementation of BIER in P4
High-level programming language to describe data plane
- Compiler maps P4 program onto programmable pipeline of target

Efficient P4-Based BIER Implementation on Tofino | IETF 115 | BIER WG
Efficient Implementation of BIER in P4 (1)

► Approach

1. Divide all ports of a switch in $k$ so-called configured port clusters $C_i$
2. Determine which configured port cluster $C_i$ requires a packet copy
   - Possibly iteratively with only a few iterations (like 1-4, not 1-32)
3. For each chosen configured port cluster $C_i$, use a suitable (static) multicast group to replicate packet to required next-hops

→ Maximal number of recirculations: $k-1$

► Example 32-port switch

- Three configured port clusters with size 11, 11, 10
- At most 2 recirculations; independent of number of next-hops
How to determine which $C_i$ requires a packet copy?

- Look at all combinations of configured port clusters $S_i$
- $C$-FBM($S_i$) is combined FBM; indicates all BFERs reachable through $S_i$
- Ternary match on BIER bitstring and complement of C-FBM $\rightarrow$ Match when result is 0
- Ordered by $|S_i|$ $\rightarrow$ Select first configured port cluster $C_j$ in $S_i$ for further processing

<table>
<thead>
<tr>
<th>Subset</th>
<th>C-FBM</th>
<th>$\neg$C-FBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1 : {C_1}$</td>
<td>00000111</td>
<td>11111000</td>
</tr>
<tr>
<td>$S_2 : {C_2}$</td>
<td>00111000</td>
<td>11000111</td>
</tr>
<tr>
<td>$S_3 : {C_3}$</td>
<td>11100000</td>
<td>00011111</td>
</tr>
<tr>
<td>$S_4 : {C_1, C_2}$</td>
<td>00111111</td>
<td>11000000</td>
</tr>
<tr>
<td>$S_5 : {C_1, C_3}$</td>
<td>11100111</td>
<td>00011000</td>
</tr>
<tr>
<td>$S_6 : {C_2, C_3}$</td>
<td>11110000</td>
<td>00000111</td>
</tr>
<tr>
<td>$S_7 : {C_1, C_2, C_3}$</td>
<td>11111111</td>
<td>00000000</td>
</tr>
</tbody>
</table>

Example BitString: 00001101
How to determine appropriate multicast group within $C_i$?
- Configure all combinations of ports within configured port cluster $C_i$ as static multicast group $H_i$
  - Robustness against change in the multicast tree, e.g., some BFERs join / leave
- C-FBM($H_i$) is combined FBM; indicates all BFERs reachable through $H_i$
- Ternary match on BIER bitstring and complement of C-FBM $\rightarrow$ Match when result is 0
- Ordered by $|H_i|$

Caveat
- $|C_i| = n$ requires $\approx 2^n$ static multicast groups

Example 32-port switch
- Three configured port clusters with size 11, 11, 10
  - $\leq 2^{11} + 2^{11} + 2^{10} = 5120$ static multicast groups

Runs at 100 Gbit/s per port on Intel™ Tofino
Optimization
Optimization (1)

► Assumptions

- Multicast traffic is not random
  - Some correlation on egress ports
- Limited number of static multicast groups available

► Idea

- Sample BIER traffic
- Store information about used ports for each BIER packet in a graph structure
- Apply clustering methods to choose configured port clusters $C_i$ such that
  - all of them cover all ports
  - few of them are needed to cover most BIER packets
Example

- BIER traffic with 4.5 next-hops on average
- Old version requires 3.5 recirculations on average (red bar)

- Highly correlated multicast traffic
  - Almost 0 recirculations with > 1024 static multicast groups

- Less correlated multicast traffic
  - ≤ 1 recirculations with > 1024 static multicast groups

(a) Traffic sampled from four generating port clusters of size 8 with different port correlation $p$. 
Mechanism to forward BIER traffic with a few iterations instead #next-hops

- Configuration: Choose configured port clusters $C_i$
- Iterative processing
  1. Select configured port cluster $C_i$
  2. Send packet to all required ports within $C_i$
- Step 2 may be achieved through pre-defined internal multicast groups
  - Does not require dynamic state
  - May be done differently depending on technology
- Max. number of iterations bounded by max. number of port clusters (~ 3 instead of 32)
- Optimized port clusters further reduce required iterations

Implemented in P4 at line rate @ Intel™ Tofino

More details
- Paper under submission: https://atlas.informatik.uni-tuebingen.de/~menth/papers/Menth22-Sub-2.pdf
- We're happy to discuss