Brief Announcement: Dynamic Forwarding Table Aggregation without Update Churn: The Case of Dependent Prefixes*

Marcin Bienkowski[1], Nadi Sarrar[2], Stefan Schmid[2], and Steve Uhlig[3]

[1] Institute of Computer Science, University of Wrocław, Poland
[2] Telekom Innovation Laboratories & TU Berlin, Germany
[3] School of EE and CS, Queen Mary University, United Kingdom

Abstract. This paper considers the problem of a route or SDN controller which manages a FIB table. The controller wants to aggregate the FIB entries as much as possible while minimizing the interactions with the FIB. We present a $O(w)$-competitive online algorithm for the aggregation of FIB tables in presence of routing updates, where w is the maximum length of an IP address. Our result is asymptotically optimal within a natural class of algorithms.

Introduction and Model. This paper studies a new online problem arising in the context of forwarding table aggregation in a router or *Software Defined Network (SDN)* switch. The *Forwarding Information Base (FIB)* contains the rules used by the router to decide, for each packet, to which port it should be forwarded; a rule is simply an (IP prefix, port) pair. We will identify ports with colors.

More specifically, any packet has a destination (IP) address which is a binary string of length w (e.g., $w = 32$ for IPv4 and $w = 128$ for IPv6). For any packet processed by the router, a decision is made on the basis of its destination IP address x using the *longest prefix match* policy: among the FIB rules $\{(p_i, c_i)\}_i$, the router chooses the longest p_i being a prefix of x, and forwards the packet to the port of color c_i. Unlike [1], we allow *dependent* prefixes, i.e., the address ranges described by prefixes stored in the FIB may be contained in each other.

In order to save memory, we let the online algorithm aggregate this table, i.e., replace the current set of rules by an *equivalent* but smaller set. In addition to reducing the number of FIB rules, an online algorithm should minimize the number of rule updates. Precisely speaking, the router consists of two parts: the *controller* (e.g., implemented on the route processor) and the *(compressed) FIB* (stored in a fast and expensive memory). The controller keeps a copy of the *uncompressed FIB (U-FIB)* and receives dynamic routing updates to this structure (that may change the color of an existing prefix). Right after such an update occurs, the controller must ensure that the U-FIB and the FIB are equivalent. To this end, the controller can insert, delete or update individual rules in the FIB, cf Fig. 1a.

* A full version of this paper can be found at [2].

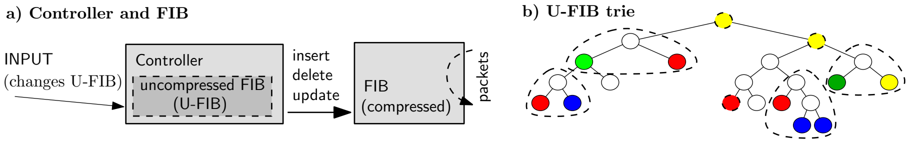

Fig. 1. *On the left:* Controller and FIB; the controller updates the rules in the FIB. This paper focuses on online algorithms for the controller. *On the right:* Example U-FIB trie. Stick boundaries are marked with dashed lines.

For presentation purposes, we represent both U-FIB and FIB as colored binary tries. These tries may contain blank nodes that do not correspond to existing rules.

Costs. We associate a fixed cost α with a change of a single rule in FIB. This paper focuses on the minimization of the sum of the total update cost and the total memory cost, where the latter is defined as the size of the FIB integrated over time.

Our Result. We present the online algorithm HiMs (Hide Invisible and Merge Sibling). HiMs is based on the concept of *sticks*: roughly speaking, a stick is a maximal part of the U-FIB trie that — if cut out of the trie — will constitute a trie of its own, with all leaves colored and all internal nodes blank. An example is given in Fig. 1b.

HiMs employs two time-delayed optimization rules: (1) If there are two siblings in a stick that are of the same color for time α, then they are removed and a rule corresponding to their parent is inserted. (2) If all colored nodes of a stick are of the same color and of the same color as their least colored ancestor in the trie (again for time α), then all these stick rules become removed from the trie. These optimizations are rolled back only when necessary to assure that the forwarding behavior of FIB is the same as that of U-FIB. A precise definition of the algorithm and its analysis are given in [2].

Theorem 1. HiMs *is $O(w)$-competitive. This is optimal in the class of all algorithms (even offline ones) that do not create dependent prefixes within a single stick. Furthermore,* HiMs *can be implemented using a data structure, whose amortized complexity for a single operation is at most $O(w)$ times the number of updates* Opt *performs in its FIB.*

References

1. M. Bienkowski and S. Schmid. Competitive FIB aggregation for independent prefixes: Online ski rental on the trie. In *Proc. of the 20th Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2013. To appear.
2. N. Sarrar, M. Bienkowski, S. Schmid, S. Uhlig, and R. Wuttke. Exploiting Locality of Churn for FIB Aggregation. In *TR 2012/12, TU Berlin*, 2012.