# Leveraging Locality for FIB Aggregation

Nadi Sarrar*, Robert Wuttke*, Stefan Schmid*, Marcin Bienkowski† and Steve Uhlig‡
*TU Berlin, {nadi,robert,stefan}@inet.tu-berlin.de
†University of Wroclaw, mbi@ii.uni.wroc.pl
‡Queen Mary, University of London, steve@eecs.qmul.ac.uk

***Abstract*—Snapshots of the Forwarding Information Base (FIB) in Internet routers can be compressed (or aggregated) to at least half of their original size, as shown by previous studies. However, the permanent stream of updates to the FIB due to routing updates complicates FIB aggregation in practice: keeping a (near-)optimally aggregated FIB in face of these routing updates is algorithmically challenging. A sensible trade-off has to be found between the aggregation gain and the complexity of handling routing updates. This paper investigates whether the spatial and temporal locality properties of routing updates conceal opportunities for improving this trade-off in online FIB aggregation.**

**Our contributions include an empirical study of the locality of updates in public Internet routing data. To facilitate this study, we design the Locality-aware FIB Aggregation (LFA) algorithm. We show, that an algorithm as simple as LFA can effectively leverage the locality of FIB churn to keep low the number of updates to the aggregated FIB, as within time periods of a few seconds or minutes, routing updates affect only a limited number of regions in the FIB.**

## I. Introduction

At a high level, Internet routers are built around a route processor which runs routing protocols and makes routing decisions, and a forwarding plane which forwards packets according to the decisions of the route processor. The crucial link between the two components is the *Forwarding Information Base* (FIB), containing the *forwarding rules*. The route processor inserts and deletes FIB entries according to its route computations. The forwarding plane uses the FIB to perform an IP destination lookup on each incoming packet. This requires the FIB to support very fast IP destination lookups so that packets can be forwarded at line-rate. In addition, the FIB needs to support frequent updates to the forwarding rules due to the churn in the BGP routing table. Finally, the number of forwarding rules that a FIB needs to keep is growing over time, putting extra pressure on the FIB memory capacity.[1] To fulfill all these requirements, FIB memory used in today's enterprise-grade to high-end routers is a highly specialized component, expensive, and power-hungry. It is also considered a main limiting factor in terms of a router's lifetime [12].

A natural and local solution to mitigate the problem — before possible long-term solutions are deployed — is the *aggregation* (or *compression*) of the FIB, i.e., the replacement of the existing set of rules by an *equivalent but smaller* set. The aggregation of FIB rules has the appealing property that it is a purely local solution, in the sense that it does not affect neighboring routers and it can be realized by the route processor through only software modifications.

While the aggregation of the FIB entries is beneficial in terms of memory, it also entails a potential overhead: As the FIB contents of a router change over time — with peaks of several hundreds to thousands of modifications inside one second [7] — the FIB aggregation algorithm needs to translate every routing update into one or more updates to the aggregated FIB. This is because when applying routing updates as they come to the aggregated FIB, forwarding consistency to the original FIB is likely to be violated [18]. Also, there is a certain cost associated with each such update as the internal FIB structures have to be updated, delaying the corresponding changes to the forwarding plane. This is particularly relevant in the context of Software Defined Networks (SDN), where the task of FIB management is decoupled from the actual forwarding device, at a likely cost of extra update processing latency. Hence, FIB aggregation algorithms should aim at limiting as much as possible the rate of updates to the aggregated FIB.

**Contributions.** In this paper, we conduct an empirical study of the temporal and spatial locality properties of routing updates in the Internet. We explore their ability to reduce the overhead in handling updates in FIB aggregation algorithms. For this purpose, we present and evaluate the *Locality-aware FIB Aggregation Algorithm* (LFA). LFA exploits the spatial and temporal locality of churn by aggregating only selected slices of the FIB (called STICKS) adaptively to amortize update complexity with the aggregation gain. We provide an empirical analysis[2] of LFA based on publicly available Internet routing data to investigate the associated trade-offs.

## II. Terminology, Related Work

**Terminology.** We consider an Internet router with a number of network interfaces, or *ports*. A *Forwarding Information Base* (FIB) is a set of *forwarding rules* used by the router for its packet forwarding operations, where each such rule is a *prefix-port* pair $(p, o)$. A *port* in this context represents all information needed for the router to forward IP packets to a given destination, which typically includes the IP address of the next-hop router. For every incoming packet the router performs a destination lookup based on the destination IP address $x$ of the packet. The destination lookup is a *longest prefix match*: Among the forwarding rules $\{(p, o)\}$, the router finds the longest $p$ being a prefix of $x$, and forwards the packet to output port $o$.

We denote the original set of forwarding rules by OT ("original table"). The OT is updated according to the routing protocols. Prior to downloading the OT into the router's FIB memory, we perform FIB aggregation: The forwarding rules of the OT are replaced by an equivalent but smaller set, denoted by AT ("aggregated table"). In this process, we require *strong forwarding correctness*, i.e., we require the forwarding behavior to remain exactly the same. On changes to the OT

[1] As of March 2014, the number of BGP routes announced in the Internet has surpassed $500,000$ entries. New forwarding rules emerge primarily because of the growth of the Internet itself, trends for advertising more specific routes, or the increasing demand for Virtual Private Networks (VPNs) [3], and the migration to IPv6 is not expected to reduce the current address space disaggregation [5].

[2] We are open to share our simulator source-code on request.

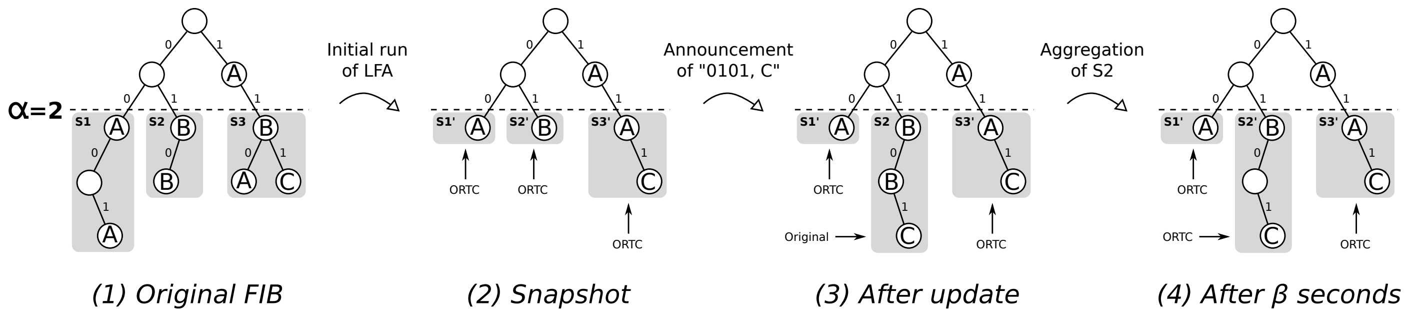


Fig. 1. Locality-aware FIB Aggregation (LFA)

due to routing updates, the AT must be updated accordingly. Updating the AT however is algorithmically challenging, as typically OT updates do not translate 1:1 into AT updates.

**Related Work.** There are known fast algorithms for optimal FIB aggregation of table snapshots, such as ORTC [6] and others [17]. However, as they do not support efficient handling of incremental updates, a re-computation of the aggregated FIB on each routing update is needed. This is computationally expensive and can lead to high churn. There are several papers that deal with this problem by proposing heuristics that balance update complexity and aggregation gain, including SMALTA [18] and FIFA [10], while others consider the entropy bounds of FIB compression [15].

With this paper we support current theoretical research in the area, which provides worst-case guarantees for the joint optimization of FIB aggregation ratio and update cost in scenarios with [1] and without [2] dependencies. Our empirical study shows, that the worst-case assumptions therein are overly conservative, and indicates that much better results can be obtained in practice.

## III. TAMING FIB UPDATE CHURN

In this section we study the locality of FIB churn based on real Internet routing data. We propose an online FIB aggregation algorithm called *Locality-aware FIB Aggregation* (LFA), whose design goal lies in enabling us to study the locality of routing updates in the context of FIB aggregation. LFA aims at aggregating stable parts of the FIB while keeping the less stable ones untouched to limit update overhead. We start by describing the LFA algorithm. Then, we present experimental results based on routing table snapshots that provide a first look at LFA's aggregation abilities. Finally, we evaluate the locality of churn under consideration of the trade-offs and parameters of LFA. From our results, we discuss and quantify the potential benefit of FIB aggregation techniques that treat churny regions of the FIB differently to those with limited churn.

### *A. LFA: Locality-aware FIB Aggregation*

LFA operates as follows. The FIB in its usual trie representation is split into subtrees (STICKs) which are aggregated only when they are considered stable: when a STICK has not been affected by updates for a pre-defined time period ($\beta$ seconds), ORTC [6] is used to optimally aggregate the STICK. On routing updates, the STICK is reverted to its original (disaggregated) representation before the update is applied. We simulate LFA on real BGP update streams and identify the trade-offs associated with its parameters $\alpha$ (spatial locality) and $\beta$ (temporal locality). In all of our simulations we verify that AT and OT are equivalent.

In LFA, the tree is split horizontally into two parts. The upper part, which we call GROUND, contains the less specific prefixes and remains untouched by LFA. Hence, as the GROUND is not subject to aggregation in LFA, routing updates to the GROUND can be applied immediately as they come (one update to the GROUND results in one update to the AT). The lower part contains the more specific prefixes and is aggregated selectively by LFA. The parameter $\alpha$ defines at which depth (prefix length) to draw the line that separates the GROUND from the more specific part of the tree. All prefixes with a prefix length $\geq \alpha$ belong to the more specific part.

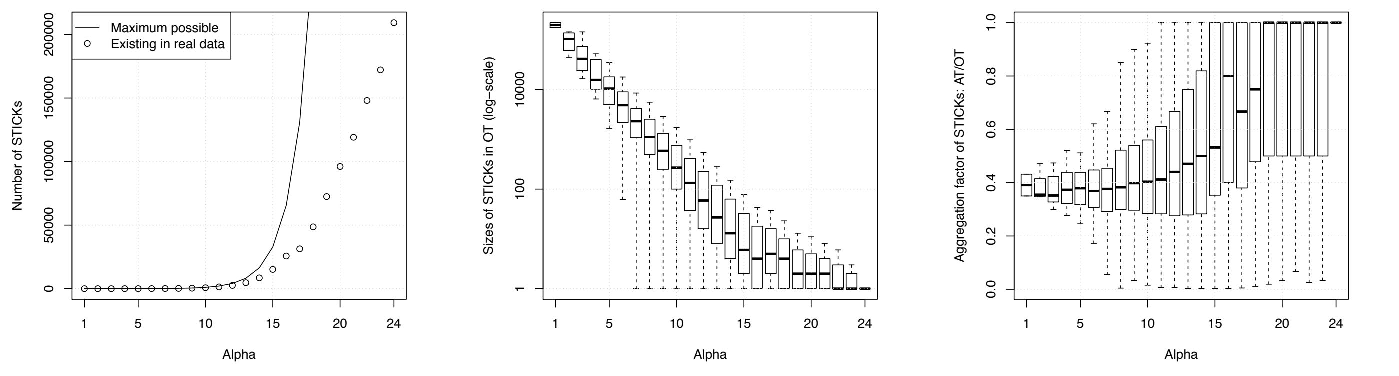
The more specific part of the tree is split vertically into subtrees, called STICKs. All the nodes with prefix length $= \alpha$ represent root nodes of individual STICKs. A STICK which has not seen any updates for a pre-defined time period is aggregated using the ORTC algorithm. In LFA, STICKs are aggregated independently from the GROUND: no next-hop information, which can change over time, is being inherited from the GROUND when aggregating a STICK. When forwarding packets, however, STICKs do depend on next-hop information from the GROUND for holes in a STICK's address space. To handle this correctly, we rely on a variant of ORTC that ensures full congruency between OT and AT, meaning that all holes in a STICK's address space are retained to allow the GROUND's entries to determine the next-hop for a packet.

The parameter $\beta$ specifies the time in number of seconds after which a STICK is aggregated in the absence of updates. For each STICK a timestamp is maintained that indicates the time of its most recent update. On incoming updates to a STICK we distinguish two cases:

1) STICK *aggregated:* In case the affected STICK is aggregated, the STICK is reverted to its non-aggregated (untouched) version prior to applying the update.
2) STICK *untouched:* Updates are applied as-is to non-aggregated STICKs.

In both cases, the STICK's update timestamp is set to the time of the update. A priority queue maintains pointers to each untouched STICK, ordered by the time of the most recent update. A timer keeps track of the tail of the queue and aggregation is applied to those STICKs that have an update timestamp $\leq$ current time - $\beta$.

Figure 1 illustrates the algorithmic components of LFA for $\alpha = 2$. The trie represents a FIB, trie levels represent prefix length starting at zero, and letters represent ports. Empty nodes do not have a corresponding entry in the FIB. The first figure highlights how $\alpha$ is used to separate the GROUND from the STICKs S1 to S3. Initially (see Figure 1 (2)), all STICKs are aggregated using ORTC while reducing the total number of prefixes from 8 to 5. In the figures, we append a prime symbol



(a) Number of existing STICKs as a function of $\alpha$. (b) Distr. of STICK sizes in OT as a function of $\alpha$. (c) Per-STICK aggregation gain as a function of $\alpha$.

Fig. 2. A first look at the impact of $\alpha$ in LFA.

to the STICK identifiers when they are aggregated, hence we now have the STICKs S1' to S3'.

Next, in Figure 1 (3), we consider an update that affects S2'. Prior to applying the update, S2' is reverted to its disaggregated form S2. After that, the update can be applied. In this example the update reflects a prefix announcement which is handled by LFA's insert procedure. Algorithm 1 provides pseudo-code for LFA's insert procedure. We leave out the delete procedure as it is similar to the insert one, except for Lines 2 and 10 call *TrieDelete()* instead of *TrieInsert()*. After S2 remains unchanged for $\beta$ seconds, S2 is aggregated again in Step (4).

**Algorithm 1** LFA-Insert($p$: prefix, $o$: next-hop)

```
1:  if $length(p) < \alpha$ then
2:      $TrieInsert(p, o)$
3:  else
4:      $S \leftarrow Stick(p)$
5:      if $IsAggregated(S)$ then
6:          $RevertToOriginal(S)$
7:      else
8:          $Dequeue(S)$
9:      end if
10:     $TrieInsert(p, o)$
11:     $SetTimestamp(S)$
12:     $Enqueue(S)$
13: end if
```

We note, that LFA introduces a certain amount of memory overhead (e.g., for the priority queues) and additional computations for performing the aggregation. However, this affects only the route controller (an embedded system in a router or a separate route server), where memory and CPU resources are cheap. The goal of this work (and several related FIB aggregation papers) is to reduce the memory needed on line-cards and to pay for it as little as possible in terms of additional churn, i.e., the amount of updates to the FIBs on the line-cards. This is particularly relevant in the context of Software Defined Networks (SDN), where the latency and capacity of the communication channel between the forwarding device and the route controller can pose significant limitations [13].

## *B. Analysis of churn locality*

LFA has been designed to facilitate studies of the locality of churn in the FIB. More specifically, LFA allows to (1) quantify the aggregatability of dependency-free[3] regions of the FIB, (2) monitor the locality of churn over time, and (3) study the trade-offs of the parameters $\alpha$ and $\beta$ of LFA.

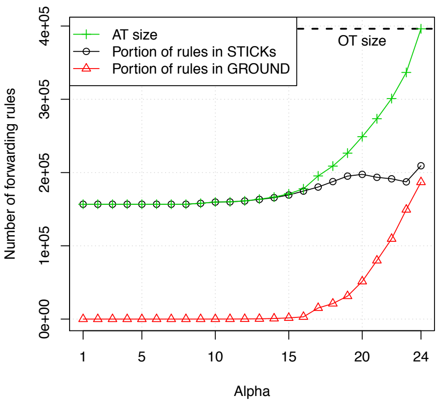


Fig. 3. Size of aggregated STICKs and GROUND as a function of $\alpha$.

**Aggregatability of STICKs.** We rely on snapshots of real routing tables to study the general aggregatability of STICKs and the dependency on $\alpha$. We obtained the routing table dumps from RouteViews[4] [14]. Due to space limitations and because the results are similar[5], we present results based on a single routing table snapshot from a large US Internet service provider. This routing table contains almost $400,000$ entries with more than 900 unique next-hop ASes, numbers so large that the results based on this dataset can be treated as upper bounds on the performance of other routers"

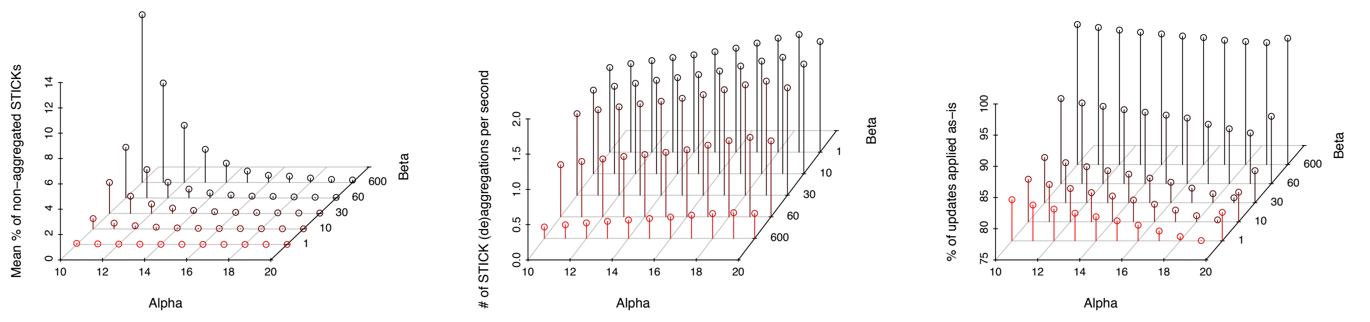
At first, in Figure 2(a), we show the number of STICKs as a function of $\alpha$. The figure compares the maximum possible number of STICKs for a given $\alpha$ with the number of existing STICKs in our snapshot. Figure 2(a) shows that the number of existing STICKs is substantially smaller than the maximum possible. This means that despite the near exhaustion of the current IPv4 address space, IPv4 FIBs are sparsely populated in terms of their filling of the tree data structure.

Figure 2(b) shows the impact of $\alpha$ on the distribution of OT STICK sizes, i.e., the numbers of prefixes in non-aggregated

[3]A dependency-free region of a FIB is a group of prefixes that does not have more specifics, but less specifics may (and typically do) exist.

[4]Due to limitations in the data we approximate ports by next-hop ASes.

[5]We ran our analyses on about 30 routing table dumps from each year between 2009 to 2012 and observed similar results.



(a) Fraction of non-aggregated STICKs.

(b) Number of STICK (dis)aggregations.

(c) Frac. of routing updates that are applied as-is.

Fig. 4. LFA trade-offs with $\alpha$ and $\beta$.

STICKs. We observe that both the average and the maximum STICK size decreases as $\alpha$ increases. For values of $\alpha$ larger than 7, the minimum STICK size goes to 1, indicating that at least one STICK contains no more than a single prefix. Figure 2(c) shows the per-STICK aggregation factor as a function of $\alpha$. For $\alpha \leq 15$, STICKs can be aggregated to half of their original size, while bigger values of $\alpha$ result in worse aggregation factors. We observe a non-monotonic behavior in Figure 2(c) for $\alpha \geq 16$. This is a result of the strong dependency of ORTC on the structure of a STICK for the efficiency of its aggregation. This dependency is more visible when STICKs are very small.

We conclude that values of $\alpha \leq 15$ will lead to good aggregation factors without incurring a high overhead for tracking and keeping the state of large numbers of STICKs, while at the same time achieving median STICK sizes of more than one. It is a necessary (but not sufficient) requirement for a STICK to be larger than one in size in order for it to be effectively aggregatable.

Figure 3 shows, as a function of $\alpha$, the total number of prefixes in the AT. We further decompose the AT size into its GROUND and STICK components. For $\alpha \leq 15$, the GROUND contributes only limited numbers of prefixes while the prefixes from the STICK components dominate the total size of the AT, which is more than 60% off from the size of the OT. This is consistent with Figure 2(c), in which we show that the aggregation gain suffers when $\alpha$ grows beyond 15. Furthermore, we observe a steep increase in the size of GROUND for $\alpha \geq 20$. At the same time, we see limited changes in the STICK sizes. As a result, the total size of the AT grows until it reaches the size of the OT (dashed line in Figure 3).

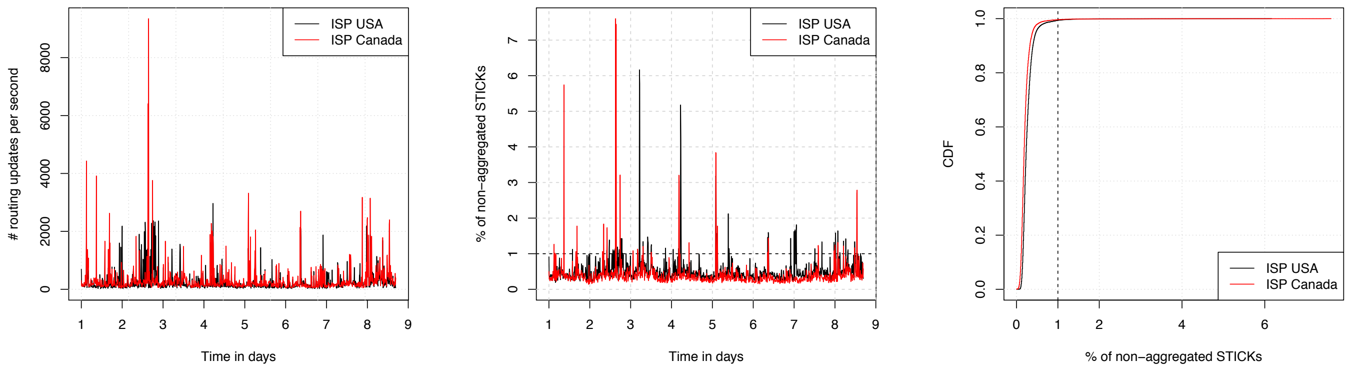
In summary, based on our analysis, a reasonable $\alpha$ in the general case of current IPv4 routing tables appears to lie below 16. The results in Figure 3 are particularly encouraging for LFA as they show that even for $\alpha$ up to 18 the total size of the FIB can be reduced by at least 50%. This gives us evidence in the approach of aggregating STICKs individually, as the achieved aggregation factors are close to those from optimal aggregation of the entire FIB [18], [6].

**Trade-offs over time.** Now that we have expectations from our analysis about the impact of $\alpha$ on the achieved aggregation factors, we now analyze the online performance of LFA under changing $\alpha$ and $\beta$. In this paper, we focus on a single dataset retrieved from a Canadian ISP router that contains more than 400,000 routing table entries. We obtain the routing table snapshot along with a stream of more than 400,000 BGP updates which cover a period of seven hours. This router has almost 200 unique next-hop ASes. We verified that the results presented are similar to those from different routers on different days. In the following results, we consider values of $\alpha$ ranging from 10 to 20, and values of $\beta$ of 1, 10, 30, 60, and 600 seconds. We chose these values of $\beta$ because they capture the scales at which BGP routing events take place [7].

In Figure 4(a) we show the fraction of STICKs over time which are not aggregated. This is a particularly important metric to consider as it provides some intuition about the locality of routing table updates. Non-aggregated STICKs represent those that have seen updates within the last $\beta$ seconds. The results indicate, that for $\alpha \geq 14$ and $\beta \leq 60s$ the fraction of non-aggregated STICKs is very low. On average, less than 0.4% of the STICKs are not aggregated (inferred by further data inspections).

Another metric to consider is the number of (dis)aggregations of STICKs over time. This metric tells us how often updates hit aggregated STICKs, requiring to disaggregate them before applying the update, and how often STICKs are aggregated after a stable period of $\beta$ seconds. In Figure 4(b) we show the average number of STICK (dis)aggregations per second as a function of $\alpha$ and $\beta$. For improved visual presentation we reverted the ordering of values on the y-axis. The results show that even for a value of $\beta$ as small as $1s$ the average number of STICK (dis)aggregations per second does not exceed 3. We also observe that this metric strongly depends on $\beta$ as the results show a steep increase when considering $\beta$ from $600s$ to $1s$.

Finally we study the impact of $\alpha$ and $\beta$ on the fraction of routing table updates which can be applied as they come. This includes all routing table updates that affect either the GROUND or non-aggregated STICKs. Figure 4(c) shows the fraction of such updates as a function of $\alpha$ and $\beta$. We observe that as $\beta$ decreases, this fraction also decreases. This is expected since smaller values of $\beta$ limit the ability of LFA to leverage update locality over time. On the other hand, the influence of $\alpha$ is non-trivial. As $\alpha$ increases, the GROUND increases, while the fraction of non-aggregated STICKs decreases (Figure 4(a)). The net effect we observe is a decrease of the number of updates that can be applied as-is. This happens because the number of updates to the GROUND increases very slowly with $\alpha$, while the fraction of non-aggregated STICKs decreases much faster



(a) Input data: number of OT updates per second (max of every 10 minute bin).

(b) Percentage of non-aggregated STICKs per second (max of every 10 minute bin).

(c) Distribution of percentages of non-aggregated STICKs per second.

Fig. 5. LFA performance over time.

with $\alpha$. The reason for this behavior is that smaller STICKs have a higher likelihood of being aggregated, as they are less likely to be affected by routing updates.

**A sensible trade-off.** We now derive a trade-off for $\alpha$ and $\beta$. Our results suggest that $\alpha$ should not be larger than 15 to achieve good aggregation gains. The results from our online experiments indicate that $\alpha$ should be $\geq 14$ to maintain a low number of non-aggregated STICKs for $\beta \leq 60s$. For $\alpha = 14$, Figure 4(a) suggests that $\beta$ should be no larger than $60s$, while Figures 4(b) and 4(c) show benefits in choosing a large value of $\beta$. In summary, our analysis suggests values of $\alpha = 15$ and $\beta = 60s$, under the churn in Internet routing tables.

While our results, which we verified using routing table data from multiple vantage points and years, provide surprisingly clear suggestions for the parameters $\alpha$ and $\beta$, this work poses the challenging question of how to algorithmically adapt these parameters over time (outside the scope of this paper).

**Performance over time.** To better understand the performance of LFA with $\alpha = 15$ and $\beta = 60s$, we now perform experiments based on more than one week worth of routing table updates. The results are shown in Figure 5 for two ISP routers, one from Canada and one from the USA. We plot the workload in Figure 5(a) as the time-series of the number of BGP updates per second. We show the maximum value for every 10 minute time interval to stress how bursty BGP updates can be. We notice several routing events which cause more than 2,000 routing table updates per second. In Figure 5(b) we plot the corresponding fraction of non-aggregated STICKs over time. Again, to give importance to the high (bad) values, we show the maximum out of every 10 minute time bin. The auto-correlation (not shown) between the original time-series used in Figures 5(a) and 5(b) exhibits the impact of $\beta$: We observe a strong correlation within time lags of 60, while larger time lags show a much smaller correlation. Finally, we show in Figure 5(c) the CDF of the fractions of non-aggregated STICKs in one second time intervals. Contrary to Figures 5(a) and 5(b) that show maximum values over 10 minute bins, Figure 5(c) provides a representative perspective on the ability of LFA to keep most of the FIB compressed over time. In more than 99% of the one second time intervals for both routers, less than 1% of the STICKs are non-aggregated. LFA is therefore able to leverage the locality in how the updates affect the FIB structure, by keeping most of it compressed.

**Putting it all together.** Our results show that there is strong locality in the routing table updates with respect to their spatial and temporal properties. This locality can be exploited by FIB aggregation algorithms such as LFA, even in the event of heavy bursts of BGP routing updates.

## IV. Discussion and Future Work

During the course of this study, two main questions arose about the underlying causes of the locality in routing updates and the performance of LFA in comparison to existing algorithms. Here, we discuss our current findings and leave more in-depth investigation for future work. While FIB aggregation is still a controversial research topic, we believe that its production use is on the horizon. Accordingly, we conclude the discussion with some notes on the impact of FIB aggregation on IP destination lookup times, and on the importance of FIB aggregation in Software Defined Networks, where more care must be taken regarding the churn in the forwarding table.

**What are the causes of locality in routing updates?**

Our evaluation of LFA shows, that routing updates exhibit locality in terms of *when* and *where* they affect the routing table. However, we did not study the underlying routing events to explain these aspects of locality.

Further inspection of our datasets reveal, that, of all next-hop ASes present in a routing table, only very few contribute most updates of a burst, e.g., two (among hundreds) next-hop ASes account for more than 50% of the updates. This suggests that routing events propagated by a very small number of ASes can cause significant bursts of updates.

With another set of experiments we take a finer grained look into the composition of update bursts by attempting to identify specific AS-AS links as main causes of such bursts. BGP updates contain the *AS-Path* attribute, which is a list of ASes representing the AS-level route to the destination. For each consecutive AS-AS pair taken from the AS-Path attributes of a burst, we check for their occurrence in all AS-Path's of that burst. We again observe a highly skewed distribution, in which 20 AS-AS pairs already cover more than 50% of all updates in a burst. Put differently, a few AS-pairs seem to be present in most BGP updates (and are localised in time), while there are multiple tens of thousands of AS-AS links in the Internet. If now only 10 to 20 of these links propagate

BGP updates for a routing event simultaneously, they are able to dominate a BGP update burst in a large share of the Internet.

Accordingly, we argue for further investigation of the actual causes of the temporal locality properties in BGP update streams. Which routing events trigger large numbers of updates in a short amount of time? Further, we want to study the causes of spatial locality by taking into account the address space use practices [5] to better understand which routing events cause a large number of updates, and how these events affect specific areas of the FIB data structure.

**How does LFA perform in comparison to SMALTA?**

Given that our evaluations shed positive light on LFA's ability to leverage routing update locality, we now position the performance of LFA in comparison to existing algorithms such as SMALTA. In a nutshell, SMALTA relies on ORTC to optimally aggregate the entire FIB, and on a routing update, the aggregated FIB is modified to comply with the update, while sacrificing optimal aggregation for some time, until ORTC is re-run. We implement SMALTA and run additional experiments. Early results indicate that SMALTA achieves aggregation factors which are in the order of 3 to 5 percentage points better than LFA, a minor difference considering the absolute aggregation factors of about 30-45%.

On the other hand, LFA causes around 10 times as many updates to the aggregated FIB when compared to SMALTA. This is due to the fact that every STICK aggregation and disaggregation can require series of updates to complete. Hence, we implement another variant of LFA which uses the SMALTA routines for handling routing updates to STICKs, instead of relying on ORTC and the costly (dis)aggregations of STICKs. In this variant, LFA outperforms SMALTA in the number of AT updates by about 7%. This motivates further research in practical locality-aware FIB aggregation schemes.

**IP destination lookup times.**

In this paper we ignored the impact that FIB aggregation may have on IP destination lookup times, because they are affected by this only to a limited extent. The state-of-the-art data structures used for destination lookups (see [11, chapter 15] and the references therein) use a variety of tree-like constructs augmented with additional information. This allows for worst-case lookup times in the order of $O(\log w)$, with $w$ being the bit-length of the address; practical implementations achieve 2-3 memory lookups on average. Additionally, little is known about proprietary data structures actually used in the routers of different vendors.

**FIB aggregation and Software Defined Networking.**

We believe that our work is particularly relevant when FIB aggregation is to be implemented in systems which have a notable delay and/or a limited communication capacity between the route processor and the forwarding engine. This is the case for very large routers, as well as for remotely controlled switches such as in large enterprise networks [8] or data centers [9]. Examples include Software Defined Networks (SDN) in general, as well as centralized control planes [4]. FIB update processing delays can lead to limitations in the number of updates per second that can be applied, as prior work has shown is the case for some existing OpenFlow implementations [13]. Our work can also be beneficial in combination with the caching-based IP router design that leverages traffic properties and OpenFlow [16].

## V. Summary

In this paper we studied the spatial and temporal locality properties of routing table updates. We proposed an online FIB aggregation algorithm, called Locality-aware FIB Aggregation (LFA), that benefits from leveraging those locality properties. We verified, through real data-based simulations, that LFA is able to keep most of the FIB aggregated through its comparably simple approach: limiting the aggregation to *stable* regions of the FIB. The lessons learned in this work, combined with related existing algorithms, provide directions toward future work on practical FIB aggregation algorithms.

## Acknowledgements

We thank Anja Feldmann (TU Berlin) for her valuable feedback. This work is supported by the Polish National Science Centre grant DEC-2013/09/B/ST6/01538, as well as the EIT ICT project 'Software-Defined Networking (SDN)'.

## References

[1] M. Bienkowski, N. Sarrar, S. Schmid, and S. Uhlig, "Competitive FIB Aggregation without Update Churn," in *Proc. ICDCS*, 2014.

[2] M. Bienkowski and S. Schmid, "Competitive fib aggregation: Online ski rental on the trie," in *Proc. SIROCCO*, 2013.

[3] T. Bu, L. Gao, and D. Towsley, "On characterizing BGP routing table growth," *Comput. Netw.*, vol. 45, 2004.

[4] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in *Proc. of NSDI*, 2005.

[5] L. Cittadini, W. Mühlbauer, S. Uhlig, R. Bush, P. Francois, and O. Maennel, "Evolution of Internet Address Space Deaggregation: Myths and Reality," *IEEE JSAC*, 2010.

[6] R. P. Draves, C. King, S. Venkatachary, and B. D. Zill, "Constructing Optimal IP Routing Tables," in *Proc. of the IEEE INFOCOM*, 1999.

[7] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "BGP Churn Evolution: A Perspective from the Core," *IEEE/ACM Trans. on Networking*, 2012.

[8] C. Kim, M. Caesar, and J. Rexford, "Floodless in seattle: a scalable ethernet architecture for large enterprises," in *Proc. ACM SIGCOMM*, 2008.

[9] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," in *Proc. of OSDI*, 2010.

[10] Y. Liu, B. Zhang, and L. Wang, "Fast Incremental FIB Aggregation," in *Proc. IEEE INFOCOM*, 2013.

[11] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers Inc., 2007.

[12] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing," RFC 4984 (Informational), IETF, 2007.

[13] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. Moore, "OFLOPS: An Open Framework for OpenFlow Switch Evaluation," in *Passive and Active Measurements Conference*, 2012.

[14] "University of Oregon Route Views Project," http://www.routeviews.org/.

[15] G. Rétvári, J. Tapolcai, A. Korösi, A. Majdán, and Z. Heszberger, "Compressing IP Forwarding Tables: Towards Entropy Bounds and Beyond," in *Proc. ACM SIGCOMM*, 2013.

[16] N. Sarrar, S. Uhlig, A. Feldmann, R. Sherwood, and X. Huang, "Leveraging Zipf's Law for Traffic Offloading," in *ACM SIGCOMM CCR*, 2012.

[17] S. Suri, T. Sandholm, and P. R. Warkhede, "Compressing Two-Dimensional Routing Tables," *Algorithmica*, 2003.

[18] Z. A. Uzmi, M. Nebel, A. Tariq, S. Jawad, R. Chen, A. Shaikh, J. Wang, and P. Francis, "SMALTA: Practical and Near-Optimal FIB Aggregation," in *Proc. of the ACM CoNEXT*, 2011.