Mille-Feuille: Putting ISP traffic under the scalpel

Olivier Tilmans [‡*], Tobias Bühler [§], Stefano Vissicchio [†], Laurent Vanbever [§]
[‡] Université catholique de Louvain, [§] ETH Zürich, [†] University College London
[‡]olivier.tilmans@uclouvain.be, [§]{buehlert, lvanbever}@ethz.ch,
[†]s.vissicchio@cs.ucl.ac.uk

ABSTRACT

For Internet Service Provider (ISP) operators, getting an accurate picture of how their network behaves is challenging. Given the traffic volumes that their networks carry and the impossibility to control end-hosts, ISP operators are typically forced to randomly sample traffic, and rely on aggregated statistics. This provides coarse-grained visibility, at a time resolution that is far from ideal (seconds or minutes).

In this paper, we present *Mille-Feuille*, a novel monitoring architecture that provides fine-grained visibility over ISP traffic. Mille-Feuille schedules activation and deactivation of traffic-mirroring rules, that are then provisioned network-wide from a central location, within milliseconds. By doing so, Mille-Feuille combines the scalability of sampling with the visibility and controllability of traffic mirroring. As a result, it supports a set of monitoring primitives, ranging from checking key performance indicators (e.g., one-way delay) for single destinations to estimating traffic matrices in sub-seconds. Our preliminary measurements on existing routers confirm that Mille-Feuille is viable in practice.

1. INTRODUCTION

ISP networks exhibit unique challenges related to network monitoring and traffic visibility. First, ISP operators do not control end-hosts, which forces them to adopt in-network solutions. Second, due to the huge traffic volumes carried by their networks and the available monitoring tools (e.g., NetFlow [2] or sFlow [14]), operators practically have to rely on random packet sampling. This approach leads to coarse-grained estimates that can take *minutes* to be collected. Third, operators lack control of which aggregated statistics are reported by each device, and when. This non-determinism (in traffic, statistics, and time) complexifies the task of building a network-wide view of the ISP forwarding state. As such, operators are often incapable of answering basic questions like: *What happens to the Google traffic entering my network?*

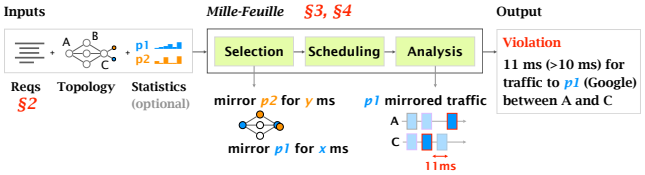


Figure 1: From high-level requirements, Mille-Feuille activates and deactivates fine-grained mirroring rules network-wide, to capture thin slices of traffic, at scale, in $\mathcal{O}(ms)$.

This work In this paper, we present Mille-Feuille[1], a fast and scalable monitoring framework based on extracting traffic samples (i.e., *slices*) which are deterministic in content, time, and space. Mille-Feuille enables ISP operators to:

- check requirements on forwarding paths: *Is the Skype traffic received by router X exiting from router Y?*
- drive traffic-engineering algorithms that require access to near real-time statistics or traffic matrices: *How much traffic towards Google is currently transiting by router X?*
- enforce Service Level Agreements, and possibly network performance: *Is the one-way delay of Skype traffic less than 20 ms, irrespectively of the ingress and egress points?*
- detect sudden network problems such as short-term congestion events, partial failures or security attacks: *Is router X dropping any packets towards Google?*

The key building block of Mille-Feuille is the ability to activate and deactivate traffic mirroring (i.e., hard-copying) for *any* destination prefix (up to a single IP address), network-wide and within milliseconds. Given a measurement objective, Mille-Feuille computes an activation/deactivation sequence, and provisions it to the routers from a central location. Mille-Feuille then uses the thin slices of mirrored traffic to build a precise view of the network-wide forwarding state.

[*]O. Tilmans is supported by a grant from F.R.S.-FNRS FRIA, and by the ARC-SDN grant from CFWB. Part of this work was done when O. Tilmans visited ETH Zürich.

HotNets-XV, November 09-10, 2016, Atlanta, GA, USA

 ISBN 978-1-4503-4661-0/16/11…$15.00
DOI: http://dx.doi.org/10.1145/3005745.3005762

[1]A French pastry meaning literally a "thousand leaves"

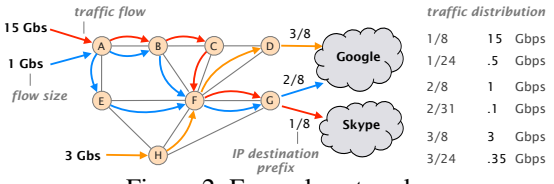


Figure 2: Example network.

By controlling what, where and when to mirror, Mille-Feuille combines the scalability benefits of sampling with the controllability and visibility advantages of traffic mirroring. Moreover, Mille-Feuille is implementable today, on current routers: traffic slices of ~15 ms can indeed be mirrored by centrally programming routers through Fibbing [17, 18].

Fig. 1 illustrates the Mille-Feuille pipeline. Mille-Feuille starts from operators' requirements, expressed in a declarative language. Then, it automatically computes a measurement campaign by combining the requirements with the physical topology and traffic statistics (if any). Mille-Feuille compiles input requirements into a distributed mirroring schedule in three steps. First, Mille-Feuille decides *what* (which prefixes) to mirror while maintaining the mirrored traffic within a given budget. Second, Mille-Feuille computes *where* (on which routers) to activate the mirroring rules, to achieve the best coverage with as few rules as possible. Third, Mille-Feuille computes *when* to mirror, and for how long. The measurement campaign is then implemented by activating and deactivating mirroring rules network-wide (in few ms). Finally, Mille-Feuille reports on which requirements are met and outputs counter-examples for requirement violations.

Novelty Most research contributions on network monitoring provide fine-grained traffic visibility in settings different from ISPs, prominently data centers (DCs). They exploit degrees of freedom that are unavailable in ISP networks, especially control of end-hosts, e.g., to collect fine-grained statistics [11]; or probe the network [5, 20]. Mille-Feuille is more general and can provide the same visibility in current ISPs, while also controlling the amount of mirrored traffic.

Contributions Our main contributions are:

- A declarative language that ISP operators can use to specify high-level monitoring requirements (§2);
- An algorithmic pipeline to compile such requirements into fine-grained mirroring rules, and schedule rule activation and deactivation over time (§3);
- A description of how to verify high-level requirements out of thin traffic slices (§4);
- An implementation of Mille-Feuille on top of unmodified routers (§5), and a preliminary evaluation (§6) assessing the feasibility of the overall framework.

Example In the paper, we consider the simple ISP network shown in Fig. 2. Routers D and G are respectively connected to Google and Skype. Google advertises 2 prefixes: 2.0.0.0/8 (to G) and 3.0.0.0/8 (to D). Skype advertises 1.0.0.0/8 (to G). Routers A and H receive traffic for Google and Skype.

pol	::=	$(m_1; \ldots; m_n)\ s^?$	Mille-Feuille Policy
m	::=	$r\ s^?$	Measurement Req.
s	::=	$\mathsf{in}(n)^?\ \mathsf{every}(n)^?\ \mathsf{using}(n)^?$	Measurement Req.
r	::=	$p\ \mathsf{within}(n)^? \mid \neg p$	Path Req.
p	::=	$\mathsf{Path}(e^+)\ \mathsf{for}(d^+)$	Path Expr.
e	::=	$id \mid *$	Node Expr.
d	::=	$netid \mid prefix$	Destination Expr.

Figure 3: Syntax of Mille-Feuille requirement language.

2. THE MILLE-FEUILLE LANGUAGE

Mille-Feuille enables to specify measuring requirements declaratively using a high-level language (Fig. 3). This language is based on the definition of (monitoring) policies. A Mille-Feuille policy is a collection of measurement *requirements*. Each requirement is composed of: *(i)* a *path requirement*, specifying how traffic should (not) be forwarded; and *(ii)* a *measurement requirement*, specifying how fast (**`in`**), how frequently (**`every`**) and using what resources (**`using`**) Mille-Feuille should verify the requirement. Path requirements capture: *(i)* where traffic is supposed to go (**`Path`**); and *(ii)* the expected delay (**`within`**). Paths are expressed as regular expressions on the underlying physical topology together with one or more destinations (**`for`**). Destinations can be v4 or v6 prefixes or an AS-level identifier (e.g., "Google").

To illustrate the main features of the language, consider a network operator who would like to assess the following four properties: *(i)* traffic for Google entering via H transits via F before leaving via D; *(ii)* traffic for Google entering via A leaves via G; *(iii)* traffic for Google entering via H never transits via C; and *(iv)* transiting traffic for Skype takes less than 20 ms, independently on where it enters or leaves. In addition, the operator would like these four properties to: *(i)* be checked every 1s; *(ii)* complete within 30 ms; and *(iii)* not generate more than 1 Gbps of mirrored traffic. These requirements are easily expressed in Mille-Feuille:

```
(  Path(H F D) for Google;
   Path(A * G) for Google;
   not(Path(H * C *)) for Google;
   Path(*) within(20ms) for Skype;
) in(30ms) every(1s) using(1Gbps)
```

3. ASSEMBLING SLICES IN A MILLE-FEUILLE

In this section, we describe how Mille-Feuille automatically translates high-level requirements into a measurement campaign consisting of quickly (de)activating fine-grained mirroring rules network-wide. To do so, the Mille-Feuille framework must answer three questions: *what* traffic to mirror (§3.1), from *where* (§3.2), and *when* and for how long (§3.3) should the traffic be mirrored.

We describe the challenges and propose algorithms to answer each question individually. We plan to assess the performance of our algorithms and compare them with different strategies in future work.

3.1 What? Computing Monitoring Targets

From a set of input queries, Mille-Feuille first computes *what to monitor*. Destinations indicated in input queries indeed map to IP ranges. For instance, in Fig. 2, Google contains all IPs in 2/8 and 3/8. In practice, any sub-prefix (up to a single IP address) can be mirrored by Mille-Feuille. We however have to decide the sub-prefixes associated to a given destination (e.g., Google) for which traffic should be mirrored to the collecting station. Those mirrored sub-prefixes must be representative of how traffic is forwarded for the corresponding destination. For example, in Fig. 2, we need to select a sub-prefix of 2/8 to verify the `Path(A * G)` requirement and a sub-prefix of 3/8 to verify the other two requirements for Google.

Challenges Selecting appropriate prefixes to mirror requires accurate estimations of the traffic attracted by sub-prefixes of the destinations specified in the input requirements. Indeed, simple assumptions such as an equal traffic split across all sub-prefixes does not work in practice as Internet traffic tend to be skewed [16]. Wrong traffic estimates can cause Mille-Feuille to mirror too much traffic, congesting the network; or not enough, not enabling it to verify requirements. For example, we cannot monitor all the prefixes (2/8 and 3/8) associated to Google in Fig. 2 without exceeding the monitoring limit of 1 Gbps stated by the query in §2.

Even with reliable traffic estimates, selecting actual sub-prefixes to monitor is still non-trivial. We indeed have to consider two conflicting objectives, that is: *(i)* minimizing the mirrored traffic to avoid network congestion and to enable to verify multiple requirements simultaneously; and *(ii)* maximizing the likelihood that packets for the chosen sub-prefixes traverse the network when they are monitored.

Our approach Mille-Feuille relies on frequent measurements to estimate traffic volumes. It aggregates results collected in previous iterations (for already-answered queries), recent data from slow monitoring sources (like NetFlow) which we assume to be anyway used for billing, and ad-hoc measurements of randomly-sampled prefixes prior to start answering the query (e.g., for initial system calibration).

We mitigate the risk to significantly exceed the monitoring budget by sharding a measurement campaign into short monitoring slices. This provides statistical guarantees that the copied traffic volume is not excessively high even if our estimations are relatively inaccurate (e.g., because of a sudden, unpredictable traffic surge).

Mille-Feuille relies on a heuristic (for time efficiency) to select sub-prefixes and distinguishes between two cases. First, for any requirement including the **not** keyword, we select the entire prefix for which the requirement must hold, so as to make sure that we catch any exception. That is, for `not(Path(H * C *)) for Google`, we will select 3/8 as monitoring target. Doing so safely (i.e., preventing unexpected mirrored traffic from overloading the networking) requires to be able to quickly stop mirroring traffic (see Sec. 6).

In contrast, for the other requirements, we define candidate sub-prefixes to mirror according to fresher measurements and we alternatively select them in a round-robin fashion. More precisely, Mille-Feuille stores and continuously updates two ordered lists. The first list A includes possible targets for a destination; the second list U contains the monitoring targets used during the last measurements. Based on these two lists, we select the sub-prefixes to mirror in the next measurement campaign, i.e., by extracting the first element from A/U. New measurements and data from slower monitoring tools can change the elements of A and their relative position in the list.

3.2 Where? Distributing Actions on Routers

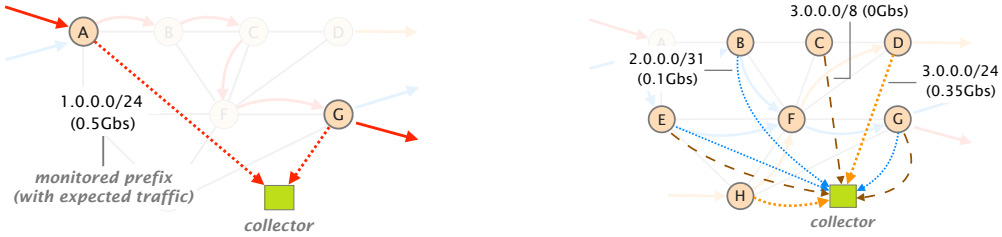
Once prefixes to be monitored are fixed, Mille-Feuille computes the actual mirroring rules and assigns them to one or more routers.

Challenges The key challenge here is the fact that many sets of mirroring rules, distributed across different nodes, can be used to answer the same query. For instance, in Fig. 2, Mille-Feuille can assess that the path from A to reach *Google* uses G as last hop, by either: *(i)* directly collecting packets from the entry and exit points (e.g., A and G); *(ii)* monitoring successors of the entry point (e.g., B and E); *(iii)* looking at the predecessors of the exit point (e.g., F); or *(iv)* a combination of the previous actions.

Naive strategies to deal with this challenge are generally inefficient and tend to often exceed the overhead limit for realistic queries. As an illustration, consider the strawman approach consisting in monitoring all the prefixes simultaneously along the paths specified in the queries. This approach would not work in Fig. 2. Indeed, we cannot monitor the entire path (HFD) taken by 3/24, carrying 0.35 Gbps of traffic, as we would end up with 1.05 Gbps of mirrored traffic (0.35 Gbps times 3 hops in the path).

In contrast, optimal approaches hardly scale with respect to computation time. For example, we can find the optimal query translation by: *(i)* enumerating all the possible actions that answer the input queries; and *(ii)* compare the respective monitoring traffic produced by each of them. However, the possible translations of a given query may result in many paths—as already exemplified for the $(A\ ^*\ G)$ path. The number of translations per query and the need to consider all their combinations would therefore make this exhaustive approach hard to scale.

Our approach Mille-Feuille builds upon a greedy heuristic to compute a spatial distribution of the mirroring rules. To assess properties on delay for a given destination, Mille-Feuille is forced to place monitoring rules on the corresponding entry and egress points. For the remaining requirements, Mille-Feuille proceeds in two steps. First, it classifies monitoring targets as heavy or light. The former are those for which requirements in the input queries cannot be verified directly, even if those prefixes are monitored alone. For instance, 3/24 is a heavy target, since mirroring the corresponding traffic



(a) $0\,\text{ms} < t < 15\,\text{ms}$. (b) $15\,\text{ms} < t < 30\,\text{ms}$.

Figure 4: Monitoring actions and traffic slices as set by Mille-Feuille during any time t in a 30 ms slot.

would violate the budget specified by the requirement query in §2. All the remaining targets are considered light. Second, Mille-Feuille allocates the monitoring actions, starting from the heavy targets. For them, it distributes actions on: *(i)* the entry and the exit points; and *(ii)* all routers adjacent to the path which packets for the target are expected to take. The former actions are needed to check that traffic for that target is actually captured during the monitoring. The latter actions are used to indirectly verify that the packets follow the expected path with no mirrored traffic. As an illustration, Mille-Feuille translates `Path(H F D)` into instructing *(i)* H and D to monitor 3/24; and *(ii)* C, E, and G to mirror traffic for 3/8. Once the heavy targets have been scheduled, Mille-Feuille distributes the monitoring actions for the light targets. For this, it relies on a greedy heuristic to spread the rules across routers as evenly as possible. This is meant to avoid monitoring bottlenecks caused by a single router that has to mirror too many destinations. In our example, Mille-Feuille instructs B and E rather than A to monitor the monitoring target for 2/8, since an action is already needed on A to assess delay for Skype traffic.

3.3 When? Scheduling Actions over Time

Even when picking optimal sub-prefixes and distributing mirroring rules across routers, the total amount of mirrored traffic is likely to exceed the allocated budget in the presence of many requirements. Fortunately, Mille-Feuille can provision, activate, and deactivate mirroring rules extremely fast, in $\mathcal{O}(ms)$. This enables Mille-Feuille to spread the mirroring campaign across time, while still maintaining low overall completion time. Monitoring actions generated for the same requirement are always scheduled at the same step.

Challenges Requirements constrain the maximum volume of monitoring traffic. This volume can be exceeded both locally and globally. Locally, requirement violations are caused by multiple actions performed by the same node. In the example of Fig. 2, Mille-Feuille has to monitor both Google and Skype traffic on G; however, it cannot schedule both actions at the same time, otherwise G will send too much traffic to the collector. In addition, the mirrored traffic at different nodes increases additively, hence potentially leading to global violation of requirements. Assume that Mille-Feuille has selected 1/24, a sub-prefix of 1/8 carrying 0.5 Gbps to monitor Skype traffic, and that it needs to monitor this traffic in two locations (A and G). In this case, it cannot monitor any other flow at the same time, since mirroring Skype traffic consumes all of the 1 Gbps monitoring budget. Finally, spreading actions too much, e.g., by sequentially mirroring one prefix in one location at the time, minimizes the overhead but also slows down completion, to the point that it can easily exceed time limits for the input queries. For instance, Mille-Feuille has at least 3 prefixes that it needs to mirror in our example, which would map to 3 monitoring time slots with the latter strategy. Assuming a minimum slice granularity of 15 ms (see §6), this would exceed the time limit of 30 ms for the query to complete.

Our approach Mille-Feuille models the problem of scheduling mirroring rules while complying with time and overhead constraints as a bin-packing problem, where: *(i)* bins represent time slots of fixed duration; *(ii)* objects are mapped to monitoring actions; and *(iii)* the weight of any object is proportional to the expected traffic received after applying the corresponding action for the entire time slot. It then uses a greedy heuristic to assign actions to time slots, iteratively trying to fill every time slot with the non-assigned actions which are heaviest from a traffic viewpoint.

Fig. 4 illustrates a possible scheduling for the above requirements. For the first 15 ms, Mille-Feuille only mirrors traffic for 1/24 in two locations for a total of 1 Gbps. In the second 15 ms, Mille-Feuille only mirrors traffic for 2/31 in three locations and for 3/24 in two locations for a total of 1 Gbps. Observe that no traffic is mirrored for 3/8 as it pertains to a negative requirement.

4. PROCESSING TRAFFIC SLICES

Traffic slices enable the Mille-Feuille controller to have complete visibility over the traffic crossing a given point. To verify high-level properties, Mille-Feuille still needs to *combine* multiple concurrent slices. Mille-Feuille distinguishes between two types of properties.

Synchronization-free properties This set regroups properties that can be verified directly by analyzing the content of slices, possibly comparing it with the content of other slices. One example is checking the validity of a forwarding path which can be done by: *(i)* locating a common packet header $\mathcal{P}$ across all the slices collected along the path; and *(ii)* observing that all subsequent packets are also following the

path. Other examples include verification (e.g., verify that a firewall is accepting/dropping properly), or analytics (e.g., compute the distribution of packets matching predicate $\mathcal{X}$).

Synchronized properties This set regroups properties that requires to estimate delays between routers. Mille-Feuille avoids requiring routers to have a synchronized clock by computing delays out of dual traces, i.e. concurrent traces where each packet arrival time has been recorded by the controller C. Let R_x and R_y be two routers. We now describe how to estimate the (one-way) delay from R_x to R_y towards prefix $\mathcal{P}$. We collect 4 traffic slices: the traffic leaving R_x towards $\mathcal{P}$, the traffic towards $\mathcal{P}$ entering R_y, and the corresponding two other slices for the reverse traffic from R_y to R_x for another prefix. We then find a packet in both slices towards R_y (resp. R_x), and compute its arrival time difference Δ_y at C across both slices (resp. Δ_x). Let δ_{ij} be the one-way delay between i and j. As δ_{xy} and δ_{yx} can differ, we have,

$$\Delta_y = \delta_{xy} + \delta_{yC} - \delta_{xC}$$
$$\Delta_x = \delta_{yx} + \delta_{xC} - \delta_{yC}$$

Mille-Feuille thus needs to estimate the delay from the routers to the controller. Without prior measurement and considering that: *(i)* the paths between the routers and the controller are symmetric (enforced by configuration); *(ii)* IGP messages are a prioritized traffic class, i.e. suffer no queuing delays, but not the mirrored traffic; *(iii)* the mirrored traffic rate is high enough; and that *(iv)* the processing time of the IGP message is constant (can be measured); Mille-Feuille can approximate the time interval between sending the IGP message enabling mirroring and receiving the first mirrored packet as being at least equal to one RTT, thus giving us an upper bound on these delays of:

$$\delta_{xC} \leqslant (t_{\text{first packet received at } R_x} - t_{\text{activation}})/2$$

Mille-Feuille can thus passively estimate the upper-bounds on the one-way delays experienced by the real traffic, and report latency violations over a path without false positives.

5. SLICING ISP TRAFFIC TODAY

This section presents how Mille-Feuille collects its traffic slices by relying on hardware-based mirroring features available in most commercial devices [1, 8]. More precisely, it dynamically programs the intra-domain routing protocol (IGP) to change the forwarding next-hop used on a per-destination basis. Those changes force specific traffic flows through a mirroring VLAN whose traffic ends up at the collector.

Network configuration We configure a monitoring VLAN that spans all intra-domain links of the network to which we allocate private IP subnets and addresses. We then advertize the VLAN in the IGP such that each link cost in the monitoring VLAN is greater than its counterpart in the original topology. Finally, we configure mirroring on all routers such that any packet *entering* the monitoring VLAN is mirrored, and its copy is encapsulated towards the monitoring controller (e.g., using Cisco's ERSPAN). As the IGP routes all traffic according to the shortest-path, the monitoring VLAN is never chosen due to its higher cost. *No traffic is thus mirrored in the initial state of the network.*

Collecting a traffic slice Assuming that we want to capture a slice of traffic towards a destination prefix p, over the link between routers R_x and R_y. First, to start mirroring the traffic, the controller programs the IGP by leveraging Fibbing [17, 18]. Fibbing enables to program arbitrary paths in the IGP (OSPF), on a per-destination basis. In this case, the controller injects a Link-State Advertisement (LSA)—an IGP message—causing R_x to prefer to route packets towards p over the monitoring VLAN (only these packets). This message is flooded in the IGP, eventually reaching R_x. As soon as the FIB of R_x is updated, R_x then mirrors and encapsulates the desired traffic. To end the slice, the controller injects another LSA which restores the previous FIB of R_x. Fig. 5a shows an example network where a mirroring rule is active.

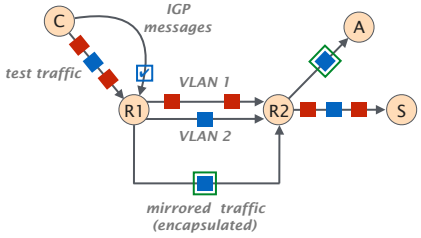
Controlling the traffic slice length Both the mirroring activation and deactivation are implemented using flooded LSAs. Since we can reasonably assume that the delay experienced by both messages to reach R_x will be the same (LSAs can be prioritized), the inter-LSA spacing between the activation and deactivation messages will be preserved by the flooding process. As the traffic slice duration is determined by the timing during which we route traffic over the monitoring VLAN, we control the slice duration by timing on the controller side the delay between the two OSPF LSAs.

Slicing traffic via the IGP is flexible and fast First, Fibbing provides a delay-insensitive way to control the slice duration. In contrast, other techniques such as CLI scripting or NETCONF are session-oriented and thus require at least one RTT for each configuration change. Their minimal slice duration is therefore governed by the delay between the controller and the target router. Second, Fibbing can (de)activate thousands of mirroring rules, for disjoint prefixes and on different routers, in $\mathcal{O}(ms)$ (see §6). Fibbing can (de)activate these rules with a single message (i.e., an OSPF LS Update containing several LSAs). Finally, Fibbing enables Mille-Feuille to drive the mirroring process via the forwarding table, that is, in hardware. Mirroring that way is therefore not impacted by control-plane constraints of existing techniques. As an illustration, our router platform (Cisco C7018) imposes a maximum of 2 active ERSPAN sessions.

6. PRELIMINARY EVALUATION

We now assess the ability of Mille-Feuille to collect *thin* slices of traffic, from *many* mirroring points simultaneously.

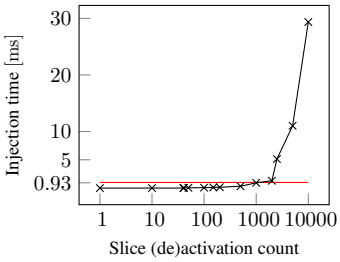
Setup Our setup (Fig. 5a) consisted of two Cisco C7018 routers ($R1$ and $R2$), a laptop running a Mille-Feuille controller (C), and two end hosts (A and S). In addition to running the controller framework, C also sends traffic at a rate of 10 packets per ms towards S. VLAN2 is the monitoring VLAN between $R1$ and $R2$, and $R1$ mirrors all traffic going



(a) Evaluation setup.

	Inter-LSA spacing	
	11 ms	20 ms
activation	4.4 ms	4.4 ms
deactivation	7.6 ms	7.2 ms
slice size	14 ms	21.2 ms

(b) Median slice sizes and (de)activation times for different Inter-LSA spacing.



(c) 1000 mirroring rules can be (de)activated at the same time.

Figure 5: In our testbed, Mille-Feuille captured traffic slices as thin as 14 ms and (de)activated 1000 mirroring rules under 1 ms.

over VLAN2 using ERSPAN and encapsulates it towards A. To mirror traffic, C uses Fibbing to route the test traffic over VLAN2.

Mille-Feuille can generate slices as thin as 14 ms We first evaluate the traffic slice sizes Mille-Feuille can achieve today. Table 5b lists the median values to activate and deactivate along with the resulting traffic slice sizes in function of the time elapsed between sending the activation and deactivation message (Inter-LSA spacing). Our results show that Fibbing can effectively (de)activate single mirroring rules while controlling the slice duration, despite being rate-limited by various OSPF timers which prevents the immediate processing of the LSA. We also confirmed that the slice duration achieved by Mille-Feuille is insensitive to the delay between the controller and the target router.

Mille-Feuille can (de)activate 1000 mirroring rules at the same time We then measured the time taken to activate many mirroring rules at once along with the control plane-overhead created in doing so. To this end, we recorded packet traces of the controller sending up to 10,000 mirroring rule activation messages through its OSPF adjacency with $R1$. As the minimal value for the timer delaying the next shortest-path computation is 1 ms, we want to batch the (de)activation of the mirroring rules as much as possible. Fig. 5c shows that we could program 1000 separate mirroring (de)activations in 0.93 ms, i.e. in a *single* SPF run—limiting the overhead on the routers to the maximum [17].

We also observed that we were able to group up to 40 mirroring rule activations in a single LSA, which reduces greatly the total amount of control-plane overhead.

7. RELATED WORK

ISP measurements Traffic sampling (e.g., via sFLOW [14]) is often used in real networks to control the amount of monitored traffic. Nevertheless, extracting samples at different routers hardly allows to track the same packets along their paths. Many approaches have been proposed to improve specific monitoring aspects, like fault detection, by analyzing router configurations (e.g., [3]) or syslogs (e.g., [4]). Ad-hoc infrastructures have also been developed to perform specific measurements inside ISP networks. For example, [13] presents a methodology to measure single-hop packet delay by using optical splitters. Similarly, in [7, 10], active probes are deployed in the Sprint IP backbone to study the impact of link failures. In contrast to Mille-Feuille, these infrastructures have a narrower scope and cannot support real-time assessment of network forwarding properties while controlling the volume of monitored traffic and the router load.

Software-Defined Networks and Data Centers Several recent monitoring contributions [19, 15, 6, 20] build upon some form of packet mirroring. They are mostly targeted to OpenFlow and/or DC networks. They typically require access to programmable switches or end hosts support so as to inject probes. A different approach is adopted in [9, 12], where high-level path queries are supported by encoding the path traversed by packets in the packet header. With respect to the works above, Mille-Feuille leverages the ability to schedule and quickly (de)activate crafted mirroring rules anywhere in the network to enable deterministic sampling. Being compatible with existing routers, it can also support abstractions similar to path queries [12] in today's networks while avoiding the extra overhead of rewriting packet headers.

8. CONCLUSIONS

This paper introduced Mille-Feuille, a novel measurement framework to provide ISP operators a complete, real-time, visibility over their network traffic. The key insight behind Mille-Feuille is to collect thin slices of traffic from multiple mirroring points by quickly (de)activating mirroring rules (within few ms). Doing so, Mille-Feuille combines the perfect visibility of traffic mirroring with the scalability benefits of traffic sampling. Mille-Feuille works with today's routers, on current hardware. Our preliminary tests show that Mille-Feuille can produce traffic slices as thin as 14 ms, and can concurrently (de)activate thousand of mirroring rules per second.

9. REFERENCES

[1] Cisco Systems. Configuring erspan, 2016. https://goo.gl/h3qaGL.

[2] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), Oct. 2004. http://www.ietf.org/rfc/rfc3954.txt.

[3] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein. A General Approach to Network Configuration Analysis. In *NSDI*, 2015.

[4] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica. X-trace: a pervasive network tracing framework. In *NSDI*, 2007.

[5] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *SIGCOMM*, 2015.

[6] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown. I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks. In *NSDI*, 2014.

[7] G. Iannaccone, C.-n. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of Link Failures in an IP Backbone. In *IMC*, 2002.

[8] Juniper Networks. Layer 2 port mirroring overview, 2014. https://goo.gl/YxgZuY.

[9] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte. Real Time Network Policy Checking Using Header Space Analysis. In *NSDI*, 2013.

[10] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. N. Chuah, Y. Ganjali, and C. Diot. Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Transactions on Networking*, 16(4):749–762, Aug 2008.

[11] M. Moshref, M. Yu, R. Govindan, and A. Vahdat. Trumpet: Timely and Precise Triggers in Data Centers. In *SIGCOMM*, 2016.

[12] S. Narayana, M. Tahmasbi, J. Rexford, and D. Walker. Compiling Path Queries. In *NSDI*, 2016.

[13] K. Papagiannaki, S. B. Moon, C. Fraleigh, P. Thiran, and C. Diot. Measurement and analysis of single-hop delay on an ip backbone network. *IEEE Journal on Selected Areas in Communications*, 21(6):908–921, 2003.

[14] P. Phaal, S. Panchen, and N. McKee. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176 (Informational), Sept. 2001.

[15] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, and R. Fonseca. Planck: Millisecond-scale Monitoring and Control for Commodity Networks. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 407–418, New York, NY, USA, 2014. ACM.

[16] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP Routing Stability of Popular Destinations. In *Proc. IMW*, 2002.

[17] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford. Central Control Over Distributed Routing. In *ACM SIGCOMM*, London, UK, August 2015.

[18] S. Vissicchio, L. Vanbever, and J. Rexford. Sweet Little Lies: Fake Topologies for Flexible Routing. In *ACM HotNets*, 2014.

[19] A. Wundsam, D. Levin, S. Seetharaman, and A. Feldmann. OFRewind: Enabling Record and Replay Troubleshooting for Networks. In *USENIX ATC*, 2011.

[20] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao, and H. Zheng. Packet-Level Telemetry in Large Datacenter Networks. In *SIGCOMM*, 2015.