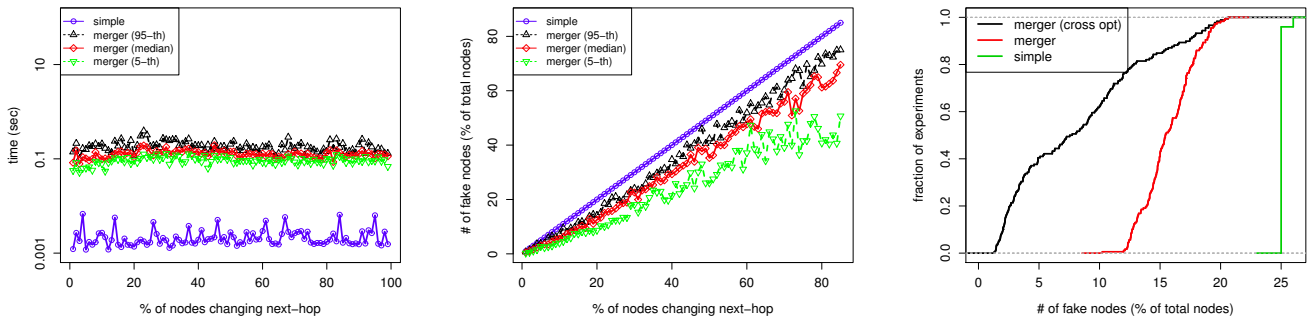


(a) Per-destination time (b) Per-destination augmentation size (c) Cross-destination reduction

Figure 8: Evaluation of our augmentation algorithms.

Fibbing can quickly program forwarding entries. In a second experiment, we measured how long it took for a router to install a growing number of forwarding entries (Table 3). We injected a growing number of fake nodes, one per destination, and measured the total installation time, by tracking the time at which the router updated the last entry in its FIB. The time to process and install one entry was constant (around 900μs), independent of the number of entries. This result is several orders of magnitude better than any OpenFlow switches currently on the market [12, 13]. Since installation of forwarding entries is distributed, routers can install their entries in parallel, meaning *Fibbing can program thousands of network-wide entries within 1 second.*

# fake nodes	installation time (s)	avg time/entry (μs)
1,000	0.89	886.00
5,000	4.46	891.40
10,000	8.96	894.50
50,000	44.74	894.78
100,000	89.50	894.98

Table 3: Programming a forwarding entry in a router with Fibbing is fast, sub 1ms.

Fibbing does not have *any* impact on routing-protocol convergence time. Finally, we compared the total time for routers to converge with and without fake nodes. We failed a link and measured the time for the last FIB entry to be updated considering two cases: (i) no lie was injected and (ii) one lie per destination was injected. Similar to the previous experiments, we repeated the measurements for a growing number of destinations and lies, between 100 and 100,000. In all our experiments, the presence of lies did not have *any* visible impact. The total convergence times with or without lies were systematically within 4ms, with the router being even faster to converge in the presence of lies in some cases.

5.2 Topology Augmentation Evaluation

We now evaluate Simple and Merger (§ 3) according to: (i) the time they take to compute an augmented topology for a given requirement, and (ii) the size of the resulting augmented topology. Results are depicted in Fig. 8. Our evaluation is based on simulation performed on realistic ISP topologies [19], whose sizes range from 80 nodes to over 300. On these topologies, we generated forwarding requirements by randomly changing the next-hop of randomly selected nodes. Destinations of requirement DAGs were also randomly generated.

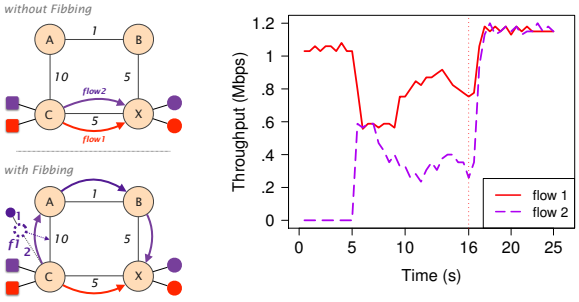
Fibbing augments network topologies within ms. Fig. 8a shows the time (on the y-axis) taken by Simple and Merger for an increasing number of nodes that must change their next-hop (on the x-axis). The plot refers to simulations we ran on the biggest Rocketfuel topology (AS1239). The time taken by Simple to compute the per-destination augmentation varies in the order of milliseconds, ranging from 0.5 ms to 8 ms. While Merger took more time (as expected), its performance is still one order of magnitude lower than the second. For both algorithms, the computation time does not vary much with the number of nodes changing their next-hops.

Merger and cross-optimization effectively reduce the size of the augmented topology. Fig. 8b plots the fake topology size (on the y-axis) when the number of nodes that have to change their next-hop increases (on the x-axis) for a single destination on all topologies. The plot shows that Merger reduces the number of fake nodes by about 25% in the average case and almost 50% in the best case. Fake topology reduction is further corroborated by our cross-destination optimization procedures (see §3.4). Fig. 8c shows a cumulative distribution function (CDF) of the topology augmentation size computed by Simple, Merger, and Merger with cross-destination optimization. The figure refers to simulations with a number of destinations varying between 1 and 100 with 26% of the nodes changing their nexthop. In more than 90% of our simulations, cross-destination optimization achieves a reduction of the augmented topology. Depending on the experi-

ment, such a reduction is up to about 10% with respect to Merger without cross-destination optimization, and 20% with respect to Simple.

5.3 Case Study

We now show the practicality of Fibbing by improving the performance of a real network consisting of four routers (Cisco 3700 running IOS v12.4(3)) connected in a square with link of 1 Mbps capacity (see Fig. 9a). In this network, we introduce two sources (bottom left) that send traffic to two destinations (bottom right) using `iperf`. The first source is introduced at time $t = 0$, the second one at time $t = 5$. OSPF weights are configured such that all traffic flows along link (C, X).



(a) Topology (b) Throughput evolution

Figure 9: Case study on how Fibbing can alleviate congestion. At $t = 16$, a fake node is added to shift one flow to the upper path in the network, increasing the total available bandwidth.

Such a network suffers from two inherent inefficiencies: (i) the upper path is never used and (ii) the two flows systematically traverse the same path, competing for bandwidth, no matter what the link weights are. Fig. 9b plots the throughput of each flow. Immediately after the introduction of the second flow at $t = 5s$, the two flows start competing for the available bandwidth. To improve network efficiency, the Fibbing controller injects a fake node f_1 connected to C and announces one destination at time $t = 16$. A few ms after the injection, we see that the throughput of both flows double as each of them now traverses a different path.

6. REACTION TO FAILURES

We now analyze Fibbing's reaction to different kinds of failures. We distinguish between network (affecting real router or router-to-router links) and controller (shutting down replicas or replica-to-router links) failures. Also, we separately deal with failures inducing network partitions and non-partitioning ones.

Fibbing quickly reacts to non-partitioning failures. Upon network failures, forwarded flows fall in one of the following three cases. First, some flows are not impacted by the failure as their pre-failure forwarding path is not disrupted. Second, flows for which no input requirements have been specified require only the IGP to establish a new path, but no action from the Fibbing controller. Reaction to failures is extremely fast in this case (sub-second even in large networks), thanks to fast convergence [20] and local fast re-route [21] features commonly supported by current IGP implementations. Third, the remaining flows are forwarded on paths modified by the Fibbing controller. They require reaction from the controller, both to remove possible blackholes or loops due to previously injected lies [14], and to avoid requirement violations due to new IGP paths. Theoretically, the total failure reaction time is equal to the sum of the notification time (for the controller to be notified of the failure), the processing time (for the controller to compute the new topology augmentation) and the IGP convergence time (for all routers to install the new lies). Our evaluation (§5) shows that the processing time is negligible, especially for the Simple algorithm. Moreover, the notification time is bounded by the IGP convergence time, as flooding is faster than re-convergence. Thus, in the worst case, the total reaction time is twice the IGP convergence time, that is, still below 2 seconds [20]. Also, in the average case, the notification time is smaller than IGP convergence, because the controller is notified about the failure before all other routers complete convergence; hence, the controller injects new lies *during* the IGP convergence, and the total failure reaction time is slightly higher than IGP convergence without Fibbing.

In addition, if one or more controller replicas fail but others are running, we have no impact on forwarded flows, unless the failed replica is the active one and some of its injected lies expire before the new active replica is elected. Even in the latter case, the new active replica is quickly elected, in a time which is approximately equal to the detection and flooding of the failure event by the IGP. The short election time makes it unlikely that lies expire before the new active replica is elected, and limits the period with possible disruptions.

Fibbing can implement both fail-open and fail-close semantics to deal with partitions. Even if unlikely, catastrophic events like a simultaneous failure of all the controller replicas or network partitions may happen. As for any centralized solution, a major risk in those cases is to leave the network uncontrolled. This happens, for example, if some routers are not reachable by a controller replica after a network partition. With respect to pure SDN solutions, Fibbing has the additional possibility to delegate control to the underlying IGP. This way, Fibbing can implement both the fail-open or fail-close semantics, on a per-destination basis. For non-critical (optimization) requirements like traffic engineering ones, the corresponding destinations can be injected in the IGP, so that connectivity can be preserved as long as the partition leaves at least one source-destination path. For stringent requirements like security ones (*e.g.*, firewall traversal), Fibbing can im-

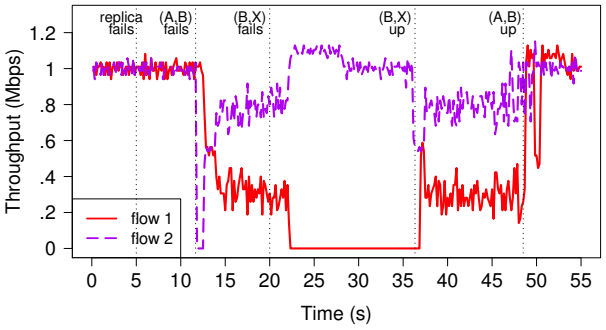


Figure 10: Case study on how Fibbing reacts to failures, and can successfully implement fail-close (flow 1) and fail-open (flow 2) semantics.

plement fail-close semantics by not announcing the corresponding destinations in the IGP. As such, the corresponding flows stop to be forwarded in the absence of the controller. To quickly reach this configuration, we can set a low validity time of the injected lies, making them rapidly expire if not refreshed. This then comes at the cost of additional control-plane overhead.

We confirmed Fibbing resilience. We consider again the topology in Figure 9a, and we connect two controller replicas respectively to routers A and B. The active replica is initially the one connected to A. We assume a strict policy on the red flow forcing it to cross the link (C, X). We then configure a fail-close semantics to it, and a fail-open to the other flow. Starting from a state in which both replicas and all links are up, we successively fail (i) the active replica at time $t = 5$; (ii) link (A, B) at $t = 12$; and (iii) link (B, X) at $t = 20$. Finally, we re-establish both failed links, one at the time (at $t = 36$ and $t = 48$).

The results of this experiment, collected via `iperf`, are reported in Figure 10. Concretely, the *failure of the active replica has no impact* on the forwarded flows. Indeed, the initially passive replica (connected to B) quickly detects the failure of the other replica, and start refreshing the injected lies by the failed controller. When (A, B) fails, the active replica needs to remove the fake node $f1$: Since the physical path (C, A, B, X) is not available anymore, this fake node is creating a loop between C and A for the violet flow. Upon failure detection, the controller then sends the LSA to remove $f1$, re-establishing the connectivity for the disrupted flow in approximately $1s$. Note that this time can be lowered by relying on fast failure detection mechanisms (like BFD). When (B, X) also fails, we create a partition that makes it impossible for the running replica to interact with routers A, C, and X. After about $1s$, the injected lies disappear, because they are not refreshed anymore by any controller. Consistent with the configured failure semantics, the red flow is blackholed (to avoid the IGP routing it over policy-violating paths) while the violet flow keeps using the IGP shortest path. Finally, re-adding the failed links allows the running replica to re-take control of the network: it re-builds a (safe) path for the red flow upon (B, X) restoration, and re-optimizes the distribution of both flow over the available paths when (A, B) is restored.

7. FREQUENTLY ASKED QUESTIONS

We now provide answers to high-level concerns often raised against Fibbing. Since empirical analyses are hardly applicable to those concerns (*e.g.*, debuggability), we describe qualitative considerations.

Is Fibbing a long-term solution? *Yes.* We believe Fibbing is here to stay. In the short run, Fibbing offers programmability and is easy to deploy, at very little cost. A network that ultimately needs even greater flexibility could deploy finer-grained SDN functionality at the edge, and solutions like Fibbing in the core, as advocated by major industry [22] and academic actors [10, 23]. By combining the best of centralized and distributed routing, Fibbing fits the needs of the network core (flexibility, robustness, low overhead) better than current forwarding paradigms.

Does Fibbing make networks harder to debug? *No.* Fibbing relies on "tried and true" protocols. This has several implications. First, Fibbing routing matches the current mental model of operators, a major advantage with respect to other SDN proposals. Moreover, Fibbing is compatible with any existing management, monitoring, and debugging tools. Finally, the Fibbing controller can expose a higher-level interface for debugging, including a mapping between the injected lies and their usage (matched requirements and how).

Does Fibbing sum the complexities of centralized and distributed approaches? *No.* Fibbing uses the underlying IGP in a very simple way. The IGP output is easy to predict and provides the controller with a powerful API to program routers. As a result, the design of the Fibbing controller is significantly simpler than for existing SDN controllers (*e.g.*, [24, 25, 26, 27]) since heavy tasks such as path computation and topology maintenance are offloaded to the routers. Even basic primitives for controller replication and replica consistency are mainly delegated to current distributed routing protocols (see §6).

Does Fibbing impact security? *No.* The lies introduced by the Fibbing controller can easily be authenticated, *e.g.*, using MD5-based authentication [28, 29].

Since Fibbing can only program loop-free paths, can it support middleboxes chaining? *Partially.* Forwarding loops can be encountered when steering traffic through a chain of middleboxes (*e.g.*, [30] and [31]). These requirements *can* be satisfied in Fibbing with local support from routers to break the loops. For instance, a router could match on the input interface in

addition to the destination IP address using policy-based routing, a feature widely available on existing routers [32, 33] and provisioned centrally using BGP flowspec [34, 35]. Alternatively, middlebox traffic steering could be implemented through SDN functionality at the network edge, while still using Fibbing in the core.

8. RELATED WORK

Fibbing contributes to the larger debate about centralized and distributed control over routing by identifying a new point in the design space.

Centralized configuration of distributed routing protocols: A centralized management system can perform traffic engineering by optimizing the link weights in link-state routing protocols [36, 37]. Fibbing is more general, since it can implement *any* forwarding paths by injecting fake nodes and links into the link-state routing topology. The extra flexibility enables even better load balancing, as well as a wider range of functionality.

Centralized control using existing routing protocols as a control channel: RCP [11] is a logically-centralized platform that uses BGP to install forwarding entries into routers. RCP must install forwarding entries one-by-one, on each device. In contrast, Fibbing can adapt the forwarding behavior of many routers at once, with little input (*e.g.*, one fake node), and let them compute their own forwarding entries.

Centralized control over the routing/forwarding tables: In SDN, a central controller installs packet-processing rules directly in the switches, possibly reacting to the reception of specific packets. While more flexible (*e.g.*, enabling stateful control logic) than Fibbing, SDN requires updating the switch-level rules one-by-one, and forgoes the scalability and reliability benefits of distributed routing. Recently, the IETF developed I2RS [38] which offers a new management interface for centralized updates the routing information bases (RIBs) in the routers. Still, I2RS must push RIB entries individually to each router.

The Fibbing language for expressing requirements is similar in spirit to Merlin [39], but the mechanism for satisfying the requirements (*i.e.*, fake nodes/links) is entirely different. Our main contributions are the Fibbing techniques and algorithms, not the language.

For the networks which require the extra flexibility provided by OpenFlow, Fibbing helps during the transition by providing access to the FIBs of legacy routers to any SDN controller [40]. This contrasts to techniques like Panopticon [41], where programmability is only available in the SDN-enabled parts of the network.

9. CONCLUSIONS

The advent of SDN makes it clear that network operators want their networks to be more programmable and easier to manage centrally. In this paper, we show how Fibbing can achieve those objectives, by centrally and automatically controlling forwarding without forgoing the benefits of distributed routing protocols. Fibbing is expressive, scalable, and works with existing routers. In future work, we plan to look at extensions of IGP protocols (*e.g.*, for source-destination routing [42] or network service header awareness) to enable finer-grained control via Fibbing. Abstractly, Fibbing shows how centralized and distributed approaches can be profitably combined. We believe that new research can further explore this direction, for example, investigating an alternative division of tasks between centralized and distributed network components.

Acknowledgements

We are grateful to SIGCOMM anonymous reviewers and our shepherd, Teemu Koponen, for insightful comments. We thank Jo Segaert from BELNET and Dave Ward, Clarence Filsfils and Kris Michielsen from Cisco Systems for their support in testing Fibbing on real routers. This work has been partially supported by the EC Seventh Framework Programme (FP7/2007-2013) grant no. 317647 (Leone) and by the ARC grant 13/18-054 from Communauté française de Belgique.

10. REFERENCES

[1] D. Awduche *et al.*, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, 2001.

[2] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 69–74, 2008.

[3] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," RFC 4655, 2006.

[4] C. Filsfils *et al.*, "Segment Routing Architecture," Internet Draft, 2014.

[5] B. Clouston and B. Moore, "Definitions of Managed Objects for HPR using SMIv2," RFC 2238, 1997.

[6] D. Oran, "OSI IS-IS Intra-domain Routing Protocol," RFC 1142, 1990.

[7] A. Pathak, M. Zhang, Y. C. Hu, R. Mahajan, and D. A. Maltz, "Latency inflation with MPLS-based traffic engineering," in *IMC*, 2011.

[8] S. Jain *et al.*, "B4: Experience with a Globally-Deployed Software Defined WAN," in *SIGCOMM*, 2013.

[9] C.-Y. Hong *et al.*, "Achieving High Utilization with Software-Driven WAN," in *SIGCOMM*, 2013.

[10] M. Casado *et al.*, "Fabric: A retrospective on evolving sdn," in *HotSDN*, 2012.

[11] M. Caesar *et al.*, "Design and implementation of a routing control platform," in *NSDI*, 2005.

[12] X. Jin *et al.*, "Dynamic scheduling of network updates," in *SIGCOMM*, 2014.

[13] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An Open Framework for Openflow Switch Evaluation," in *PAM*, 2012.

[14] S. Vissicchio, L. Vanbever, and J. Rexford, "Sweet little lies: Fake topologies for flexible routing," in *Hotnets*, 2014.

[15] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central Control over Distributed Routing (Extended Version)," Technical Report, 2015.

[16] L. Vanbever, S. Vissicchio, C. Pelsser, P. Francois, and O. Bonaventure, "Seamless Network-Wide IGP Migrations," in *SIGCOMM*, 2011.

[17] "Quagga routing suite," www.nongnu.org/quagga.

[18] J. Moy, "OSPF Version 2," RFC 2328, Apr. 1998.

[19] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *SIGCOMM*, 2002.

[20] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving Sub-second IGP Convergence in Large IP Networks," *ACM SIGCOMM CCR*, vol. 35, no. 3, 2005.

[21] C. Filsfils, P. Francois, M. Shand, B. Decraene, J. Uttaro, N. Leymann, and M. Horneffer, "Loop-Free Alternate (LFA) Applicability in Service Provider (SP) Networks," RFC 6571, 2012.

[22] T. Koponen *et al.*, "Network Virtualization in Multi-tenant Datacenters," in *NSDI*, 2014.

[23] "Time for an SDN Sequel? Scott Shenker Preaches SDN Version 2," www.sdxcentral.com/articles/news/scott-shenker-preaches-revised-sdn-sdnv2/2014/10/.

[24] "ONOS: Open Network Operating System," http://onosproject.org/.

[25] T. Koponen *et al.*, "Onix: A distributed control platform for large-scale production networks," in *OSDI*, 2010.

[26] "Project Floodlight," http://www.projectfloodlight.org/floodlight/.

[27] N. Foster *et al.*, "Languages for software-defined networks," *IEEE Comm. Mag.*, 2013.

[28] "Cisco OSPF MD5 Authentication," http://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/13697-25.html.

[29] "Juniper OSPF MD5 Authentication," http://www.juniper.net/documentation/en_US/junos14.2/topics/topic-map/ospf-authentication.html.

[30] Z. A. Qazi *et al.*, "Simple-fying middlebox policy enforcement using sdn," in *SIGCOMM*, 2013.

[31] S. K. Fayazbakhsh *et al.*, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *NSDI*, 2014.

[32] "Cisco. Configuring Policy-Based Routing," http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfpbr.html.

[33] "Juniper. Configuring Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address," http://www.juniper.net/techpubs/en_US/junos12.2/topics/topic-map/filter-based-forwarding-policy-based-routing.html.

[34] "Cisco. Implementing BGP Flowspec," http://www.cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k_r5-2/routing/configuration/guide/b_routing_cg52xasr9k/b_routing_cg52xasr9k_chapter_011.html.

[35] "Juniper. Enabling BGP to Carry Flow-Specification Routes," https://www.juniper.net/documentation/en_US/junos12.3/topics/example/routing-bgp-flow-specification-routes.html.

[36] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *INFOCOM*, 2000.

[37] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Comm. Mag.*, vol. 40, no. 10, pp. 118–124, 2002.

[38] A. Atlas, J. Halpern, S. Hares, and D. Ward, "An Architecture for the Interface to the Routing System," Internet Draft, 2013.

[39] R. Soulé *et al.*, "Merlin: A language for provisioning network resources," in *CoNEXT*, 2014.

[40] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM CCR*, vol. 44, no. 2, pp. 70–75, 2014.

[41] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks," in *USENIX ATC*, 2014.

[42] F. Baker, "IPv6 Source/Destination Routing using OSPFv3," Internet Draft, 2013.