Routing and State Distribution Trade-Offs in SDN

Nadi Sarrar
TU Berlin / T-Labs
nadi@net.t-labs.tu-berlin.de

Dan Levin
TU Berlin / T-Labs
dan@net.t-labs.tu-berlin.de

Anja Feldmann
TU Berlin / T-Labs
anja@net.t-labs.tu-berlin.de

I. Introduction

Software Defined Networking (SDN) introduces programability into the network forwarding plane and simultaneously enables architectural refactoring of the routing control plane. In this work, we present two complementary studies leveraging the opportunities presented by SDN. First, in the context of internet router architectures in Section II, we propose leveraging OpenFlow forwarding information base (FIB) programability to employ routing table aggregation techniques to reduce FIB memory requirements. Second, in Section III, we investigate a fundamental network design choice introduced by SDN: How centralized vs. distributed to build the routing control plane and what are the crucial considerations when designing a distributed SDN control plane application?

II. Alternative Router Designs

In [4], Sarrar et al. propose a novel router architecture to combine the flexibility of software routers with the forwarding performance of SDN-enabled hardware switches. They leverage the Zipf-like properties of Internet traffic to handle most traffic in the fast switch hardware, using only a small number of "heavy hitter" IP prefix forwarding entries, which collectively match a significant fraction of the total traffic. The heavy hitter selection strategy TFO (Traffic-aware Flow Offloading) compiles and updates the set of heavy hitters over time, such that the dynamics in the rank of prefixes are considered. The non-offloaded traffic is handled at the controller, i.e., the PC that runs the software router. The objectives are (1) achieving a high traffic offloading ratio, and (2) keeping the number of updates to the switch's FIB low.

We now raise the question whether FIB aggregation techniques can be implemented into such a system to further improve on its objectives. More specifically, we analyze the characteristics of FIB aggregation and its impact on the data plane in the context of a traffic offloading system which aims at offloading most of the traffic by focussing on the few most heavily used entries in the FIB of an Internet router. We want to find empirical answers, based on ISP as well as IXP traffic traces, to the following four key questions.

1. Does the number of next-hop routers present in a FIB have an impact on its aggregatability?

Two FIB entries – IP prefixes – can be merged without violating routing consistency, if (1) there exists a prefix that exactly covers their combined address space, and (2) the two original FIB entries share the same next-hop. These rules can be relaxed, see [1]. Intuitively, the larger the number of different next-hops present in a FIB, the worse the aggregation ratio. We perform experiments on real routing tables taken from ISP access and backbone routers (tens of next-hops) and routers at an IXP (hundreds of next-hops). Our preliminary results show that, with a provably optimal routing table aggregation algorithm, we can shrink FIBs to at least 60% of their original size, for all three router locations.

2. What is the right trade-off between the compression gain and frequency of changes to the aggregated FIB?

In contrast to the offline algorithm used for static compression only, real-world systems such as our traffic offloading system require an *online aggregation algorithm* that also takes into account the number of changes to the aggregated table on routing table updates, as those updates typically impact the data-plane. To achieve this, both existing online aggregation algorithms (SMALTA [5]) as well as our own algorithm seek to strike the right balance between optimal aggregation and frequency of updates. We will perform simulations with real Internet traffic data to explore this trade-off and use the insight to identify design criteria for an optimized algorithm.

3. How does the effectiveness of the traffic offloading system change under aggregation?

The offloading ratio can potentially be improved by aggregation. Each single offloaded routing table entry will comprise multiples of the original entries and thus have a bigger share in traffic volume. We will perform simulations based on real Internet traffic traces to answer questions like (1) how compressible are the offloaded, top-volume FIB entries, and (2) how much more traffic can be offloaded with the help of prior aggregation? Preliminary results based on IXP data suggest that the fraction of offloaded traffic can be increased by more than 10%, when offloading 2,000 FIB entries[1].

4. How does the number of updates to the offloaded FIB entries change under aggregation?

Depending on the specific behavior of any aggregation algorithm, the offloaded FIB entries will likely be aggregated in various different ways. This in turn leads to different numbers of modifications to the offloaded entries over time (*churn*), depending on the algorithm used. We are the first to study the churn of aggregated routing tables specifically for those entries which carry most of the traffic, i.e., the ones that the traffic offloading system uses to offload traffic.

[1]Based on a snapshot of the IXP's route server's FIB and a 1-day trace of sFlow samples, for bin-optimal and TFO strategies [4].

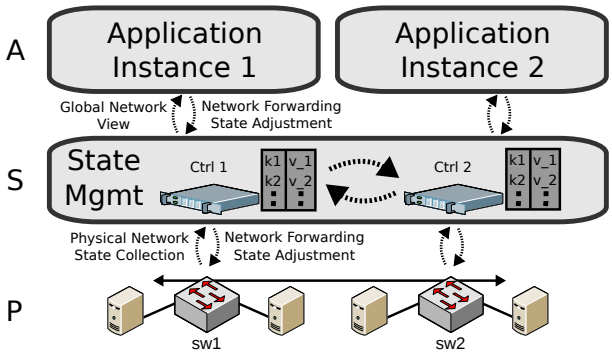

Fig. 1. SDN state distribution and management conceptualized in Layers: (A)pplication, (S)tate Management, (P)hysical Network

III. SDN State Distribution Trade-offs

One of the key features enabled through the SDN control- and data-plane decoupling is the ability to design and reason about the network control plane as a *centrally* controlled application operating on a global network view (GNV) as its input. Essentially, SDN enables network control logic to be designed and operated as though it were a centralized application, rather than a distributed system – logically centralized [2]. Thus, SDN designers now face new choices, in particular: How centralized or distributed should the network control plane be. Fully *physically* centralized control is inadequate because it limits *(i)* responsiveness, *(ii)* reliability, and *(iii)* scalability. Thus, designers resort to a physically distributed control plane, on which a logically centralized control plane operates. Physically distributed control plane state can impact the performance and correctness of a control application logic designed to operate as though it were centralized. Consequently we ask, when the underlying distributed control plane state leads to inconsistency in the global network view, how much does the network performance and correctness suffer?

A. Approach

To approach this question, we begin with a systematic characterization of the state exchange points, illustrated in Figure 1 in a multi-domain SDN. We then identify two key state distribution trade-offs that arise when a logically centralized controller application is run on top of physically distributed control plane state: *(i)* The trade-off between control application performance (optimality) and correctness and SDN state synchronization overhead and *(ii)* application logic complexity vs. robustness to inconsistency in the underlying distributed SDN state.

Each controller maintains a *Network Information Base* (NIB) data structure, which is a view of the global network state presented to an application. For instance, the NIB contents presented to a network load-balancing SDN control application would include at least link capacity and utilization state. The NIB at each controller is periodically and independently updated with state collected from the physical network (e.g., through port counters or flow-level statistics gathering). Additionally, controllers synchronize their NIB state among themselves in order to disseminate their domain state to other controller domains.

Different manners of distributed, replicated storage models may be used to realize the Network Operating System (NOS) state distribution and management, including transactional databases, distributed hash tables, and partial-quorum mechanisms [3]. One key property of any NOS state distribution approach is the degree of state consistency achieved – strong (e.g., via transactional storage) vs. eventual consistency. While a strongly consistent NOS will never present inconsistent NIB state to an application (i.e., no two application instances will see a different global network view), it will impose a bottleneck to the rate at which NOS state can be updated. Eventually consistent approaches however will lead to inconsistency – different global network view being presented to different application instances.

Trade-off #1 arises when a "logically centralized" control application is ignorant to the underlying distributed NOS state. Inconsistent updates to the physical network forwarding state may result in routing loops, unoptimal load-balancing, and other undesired application-specific behavior. The cost to reducing stale state in the global network view entails higher rates of control synchronization and communication overhead.

The degree to which the control application logic is more or less *aware* of the distributed nature of the underlying global network view constitutes **trade-off #2** between application logic complexity and robustness to stale NOS state. An application which is aware of underlying distributed NOS state can take measures to separate and compare the inter-domain global network view with its own local domain view, and avoid taking action based solely on stale input.

B. Trade-off Simulation

We simulate these trade-offs in the context of an existing SDN network flow-based load-balancing application. We compare two different control application approaches which operate on distributed SDN state: A simple approach that is ignorant to potential inconsistency in the global network view, and a more complex approach that considers the potential inconsistency in its network view when making a load-balancing decision. In our specific simulation scenario, initial results demonstrate that GNV inconsistency significantly degrades the performance of our network load-balancing application which is naive to the underlying distributed SDN state. Alternately, the more complex application state management approach is more robust to GNV inconsistency.

References

[1] R. Draves, C. King, S. Venkatachary, and B. Zill. Constructing Optimal IP Routing Tables. In *INFOCOM*, 1999.

[2] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann. Logically centralized? state distribution tradeoffs in software defined networks. In *HotSDN Workshop*, August 2012.

[3] Y. Saito and M. Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81, Mar. 2005.

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

[5] 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 *ACM CoNEXT*, 2011.