SDX: A Software Defined Internet Exchange

Why SDX?

BGP’s peering decisions are constrained by destination IP and obscure BGP specific parameters. SDX not only simplifies but allows for fine grained peering configurations based on traffic belonging to certain application, from certain sender or to control end-to-end paths. Some motivating cases from the paper are given below

- Video traffic constitutes a significant portion of overall traffic volume and the solution available for application specific peering is to use VRF(Virtual Routing and Forwarding tables)\(^1\) for each traffic class and direct traffic of that class to its corresponding VRF.

- Existing solutions to control inbound traffic are AS path prepending, communities and selective advertisement by recipient. SDN can perform inbound traffic engineering naturally using src IP and port.

- Currently wide area server load balancing is done using DNS. DNS caching slows down load shifting in case of server failure. With SDX, content providers could assign a single anycast\(^2\) IP for a service which is dynamically rewritten in the middle of network (at IXP).

- Steering traffic to middleboxes involves manipulating underlying routing protocol to hijack offending traffic. But alongside much more normal traffic is also affected. With SDX, desired subsets are redirected to middleboxes

SDN can be used as a tool owing to its enhanced packet processing capabilities. The motivating cases in this paper are mere(simpler) alternatives to what BGP accomplishes in a convoluted way and are not extra ordinarily compelling for IXP’s or service providers but as is discussed in later work in the series, SDX establishes interconnection for software defined infrastructure and can serve as a platform for SDN services and network functions.

System Design

To realize a software-defined IXP, what is needed?

---

\(^1\)VRF: Multiple virtual router instances running on a single physical box. Each instance with its own separate interfaces, subnets, RIB, FIB and routing protocols (similar to VLANs)

\(^2\)Anycast addressing routes datagrams to a single member of a group of potential receivers that are all identified by the same destination address. This is a one-to-nearest association.
• Programming abstractions for participants
• A run-time to coordinate between policies of different participants and interact with BGP

What are the challenges?
• correctness
• scalability
• flexibility
• incremental deployment

To provide flexibility as well as isolation, each participating network is given the illusion of its own virtual SDN switch connecting its border router to each of its peers ASes. Pyretic is used to write policies. Participating networks having physical presence at the IXP can have both inbound as well as outbound policies. Networks not physically present at IXP can only specify inbound policies for wide area load balancing.

Participating networks exchange routes with SDX route server. For incremental deployment, networks which are happy with BGP as is do not need to provide any policies and SDX route server will act as an ordinary route server. For correctness, SDX applies participant policies only on advertised routes (subset of traffic allowed by BGP) by combining src(outbound) policies with dest(inbound) policies and augmenting src and dest switch ports on a specific route. A combined policy is translated into forwarding rules by Pyretic. Pyretic supports topology abstraction and so can handle SDX consisting of multiple physical switches.

Data-plane scalability: Large IXPs host several hundred participants and considering fine-grained policies from all of those participants would result in millions of rules (rule installation time, rule capacity at the switches). To avoid that prefixes are grouped into equivalent classes based on forwarding behavior. Prefix groups are computed using Minimum Disjoint Subset algorithm and a two stage FIB table is used for each participant. First table assigns VMAC tag learnt from SDX router server. Second table performs forwarding action based on VMAC.

This essentially reduces forwarding rules in 2nd table

Control-plane scalability: Policy compilation is optimized for efficient computation. Instead of using any complex algorithm, insights about underlying phenomenon are used to optimize To optimize initial compilation, 1) only compose policies among participants that exchange traffic, 2) Do not apply parallel composition to disjoint sets of traffic, 3) memorize intermediate compilation results to prevent recompiling same policy.

To optimize incremental update at SDX caused by underlying BGP route changes, some observations about BGP updates were utilized; 1) Prefixes that are likely to appear tend to be able and do not see frequent updates, 2) most BGP update bursts (~75%) affect a small number of prefixes (≤3), 3) BGP update burst are separated by large periods with no changes, enabling quick, sub-optimal reaction followed by background re-optimization.
Implementation and Evaluation

Deployed both a prototype as well as on Mininet. Virtual ASes, connected to OVS, establish connectivity to Internet via "Transit Portal" (at University of Wisconsin and Clemson University). Two servers in Amazon cloud for load balancing experiments.

For experiments, policies and topologies are derived from the "characteristics" of AMS-IX, LINX, DEC-IX and traffic characteristics are derived from advertised prefixes. So all input are modeled and not real data. It is assumed that only following install custom policies: 1) Top 15% of eyeball ASes, 2) Top 5% of transit ASes, 3) Random 5% of content providers.

Forwarding Table Space in switches: As the number of prefixes to which SDX policies are applied increases, more prefixes are announced by same number of participants, thereby increasing the likelihood that advertised prefixes are part of same group and hence sub-linear increase in prefix groups. Number of forwarding rules increase linearly with number of groups.

Rule Compilation Time: Initial compilation time is of the order of several minutes and quadratically increase with increase in prefix groups. Incremental compilation takes ~100msec.

Note: Never in their performance evaluation are they considering rule installation time. Even though their observation that BGP updates are uncommon, the response time of SDX heavily relies on how quickly switches can install updated rules.