SDX: A Software-Defined Internet Exchange

Arpit Gupta

Laurent Vanbever, Muhammad Shahbaz, Sean Donovan, Brandon Schlinker, Nick Feamster, Jennifer Rexford, Scott Shenker, Russ Clark, Ethan Katz-Bassett

Georgia Tech, Princeton University, UC Berkeley, USC
The Interdomain Ecosystem is Evolving ...

Flatter and densely interconnected Internet*

*Labovitz et al., Internet Inter-Domain Traffic, SIGCOMM 2010
…But BGP is Not

• Routing **only on destination IP prefixes**
  (No customization of routes by application, sender)

• Can only influence **immediate neighbors**
  (No ability to affect path selection remotely)

• **Indirect** control over data-plane forwarding
  (Indirect mechanisms to influence path selection)

How to overcome BGP’s limitations?
SDN for Interdomain Routing

- Forwarding on *multiple header fields* (not just destination IP prefixes)
- Ability to *control entire networks* with a single software program (not just immediate neighbors)
- Direct control over data-plane forwarding (not indirect control via control-plane arcana)

How to incrementally deploy SDN for Interdomain Routing?
Deploy SDN at Internet Exchanges

- **Leverage**: SDN deployment even at single IXP can yield benefits for tens to hundreds of ISPs

- **Innovation hotbed**: Incentives to innovate as IXPs on front line of peering disputes

- **Growing in numbers**: ~100 new IXPs established in past three years*

*https://prefix.pch.net/applications/ixpdir/summary/growth/*
Background: Conventional IXPs

- AS A Router
- AS B Router
- AS C Router
- Route Server
- BGP Session
- IXP
- Switching Fabric
SDX = SDN + IXP
SDX Opens Up New Possibilities

• More flexible **business relationships**
  – Make peering decisions based on time of day, volume of traffic & nature of application

• More direct & flexible **traffic control**
  – Define fine-grained traffic engineering policies

• Better **security**
  – Prefer “more secure” routes
  – Automatically blackhole attack traffic
Use Case: Inbound Traffic Engineering

AS A Router -> SDX Controller -> C1, C2

10.0.0.0/8

SDX

AS B Router

AS C Routers
Use Case: Inbound Traffic Engineering

Incoming Data

<table>
<thead>
<tr>
<th>Incoming Traffic</th>
<th>Out Port</th>
<th>Using BGP</th>
<th>Using SDX</th>
</tr>
</thead>
<tbody>
<tr>
<td>dstport = 80</td>
<td>C1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10.0.0.0/8

AS A Router

C1

C2

AS C Routers

AS B Router
Use Case: Inbound Traffic Engineering

Incoming Data

AS A Router

C1

C2

AS B Router

Fine grained policies not possible with BGP

<table>
<thead>
<tr>
<th>Incoming Traffic</th>
<th>Out Port</th>
<th>Using BGP</th>
<th>Using SDX</th>
</tr>
</thead>
<tbody>
<tr>
<td>dstport = 80</td>
<td>C1</td>
<td>?</td>
<td></td>
</tr>
</tbody>
</table>
Use Case: Inbound Traffic Engineering

Incoming Traffic

<table>
<thead>
<tr>
<th>Incoming Traffic</th>
<th>Out Port</th>
<th>Using BGP</th>
<th>Using SDX</th>
</tr>
</thead>
<tbody>
<tr>
<td>dstport = 80</td>
<td>C1</td>
<td>?</td>
<td>match(dstport = 80) → fwd(C1)</td>
</tr>
</tbody>
</table>

Enables fine-grained traffic engineering policies
Building SDX is Challenging

• **Programming abstractions**
  – How networks define SDX policies and how are they combined together?

• **Interopperation** with BGP
  – How to provide flexibility w/o breaking global routing?

• **Scalability**
  – How to handle policies for hundreds of peers, half million prefixes and matches on multiple header fields?
Building SDX is Challenging

• Programming **abstractions**
  – How networks define SDX policies and how are they combined together?

• **Interoperation** with BGP
  – How to provide flexibility w/o breaking global routing?

• **Scalability**
  – How to handle policies for hundreds of peers, half million prefixes and matches on multiple header fields?
Directly Program the SDX Switch

Switching Fabric

match(dstport=80) → drop

match(dstport=80) → fwd(C1)

AS A & C directly program the SDX Switch
Conflicting Policies

Switching Fabric

A1

match(dstport=80) \rightarrow \text{drop}

C1

match(dstport=80) \rightarrow \text{fwd}(C1)

C2

B1

drop? C1?

How to restrict participant’s policy to traffic it sends or receives?
Virtual Switch Abstraction

Each AS writes policies for its own virtual switch

match(dstport=80) → drop

match(dstport=80) → fwd(C1)
Combining Participant’s Policies

Policy(p) = Pol_A → Pol_C
Building SDX is Challenging

• Programming abstractions
  – How networks define SDX policies and how are they combined together?

• **Interoperation** with BGP
  – How to provide flexibility w/o breaking global routing?

• Scalability
  – How to handle policies for hundreds of peers, half million prefixes and matches on multiple header fields?
Requirement: Forwarding Only Along BGP Advertised Routes

match(dstport=80) → fwd(C)
Ensure ‘p’ is not forwarded to C

\[
\text{match}(\text{dstport}=80) \rightarrow \text{fwd}(C)
\]
Solution: Policy Augmentation

\[(\text{match}(\text{dstport}=80) \land \land \text{match}(\text{dstip} = 10/8)) \rightarrow \text{fwd}(C)\]
Building SDX is Challenging

- **Programming abstractions**
  - How networks define SDX policies and how are they combined together?

- **Interoperation with BGP**
  - How to provide flexibility w/o breaking global routing?

- **Scalability**
  - How to handle policies for hundreds of peers, half million prefixes and matches on multiple header fields?
Scalability Challenges

• **Reducing Data-Plane State:** Support for all forwarding rules in (limited) switch memory

• **Reducing Control-Plane Computation:** Faster policy compilation
Scalability Challenges

• **Reducing Data-Plane State:** Support for all forwarding rules in (limited) switch memory
  millions of flow rules possible

• **Reducing Control-Plane Computation:** Faster policy compilation
  policy compilation could take hours
Reducing Data-Plane State: Observations

- Internet routing policies defined for groups of prefixes.*

- **Edge routers** can handle matches on hundreds of thousands of IP prefixes.

*Feamster et al., *Guidelines for Interdomain TE, CCR 2003*
Reducing Data-Plane State: Solution

Group prefixes with similar forwarding behavior

SDX Controller
Reducing Data-Plane State: Solution

Advertise one BGP next hop for each such prefix group
Reducing Data-Plane State: Solution

Flow rules at SDX match on BGP next hops

forward to BGP Next Hop

10/8
40/8
20/8

match on BGP Next Hop

SDX FIB

fwd(1)
fwd(2)
Reducing Data-Plane State: Solution

For hundreds of participants’ policies, few millions $\Rightarrow < 35K$ flow rules
Reducing Control-Plane Computation

• **Initial policy compilation time**
  – Leveraged domain-specific knowledge of policies
  – Hundreds of participants requires $< 15$ minutes

• **Policy recompilation time**
  – Leveraged bursty nature of BGP updates
  – Most recompilation after a BGP update $< 100$ ms
SDX Testbed

- Mininet-based Testbeds
  - Uses Transit Portal
  - Emulates edge routers

- Check out our demo
  - Application specific peering
  - Inbound traffic engineering

- Github repo: [https://github.com/sdn-ixp/sdx/](https://github.com/sdn-ixp/sdx/)
Summary

• **SDN-based exchange (SDX)** is promising for fixing Internet routing

• Solved various challenges in building a real deployable SDX

• Many open research problems, both for building and using SDX