Bringing SDN to the Internet, one exchange point at the time

Joint work with: Arpit Gupta, Muhammad Shahbaz, Sean P. Donovan, Russ Clark, Brandon Schlinker, E. Katz-Bassett, Nick Feamster, Jennifer Rexford and Scott Shenker

Laurent Vanbever
Princeton University

COS 561
November, 11 2014
BGP is notoriously inflexible and difficult to manage
BGP is notoriously inflexible and difficult to manage.

Fwd paradigm

Fwd control

Fwd influence
BGP is notoriously inflexible and difficult to manage

<table>
<thead>
<tr>
<th>BGP</th>
<th>SDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fwd paradigm</td>
<td>destination-based</td>
</tr>
<tr>
<td>Fwd control</td>
<td>indirect</td>
</tr>
<tr>
<td>Fwd influence</td>
<td>local</td>
</tr>
<tr>
<td>BGP session</td>
<td></td>
</tr>
</tbody>
</table>
SDN can enable fine-grained, flexible and direct expression of interdomain policies.

<table>
<thead>
<tr>
<th></th>
<th>BGP</th>
<th>SDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fwd paradigm</td>
<td>destination-based</td>
<td>any</td>
</tr>
<tr>
<td></td>
<td></td>
<td>source addr, ports,…</td>
</tr>
<tr>
<td>Fwd control</td>
<td>indirect</td>
<td>direct</td>
</tr>
<tr>
<td></td>
<td>configuration</td>
<td>open API (e.g., OpenFlow)</td>
</tr>
<tr>
<td>Fwd influence</td>
<td>local</td>
<td>global</td>
</tr>
<tr>
<td></td>
<td>BGP session</td>
<td>remote controller control</td>
</tr>
</tbody>
</table>
How do you deploy SDN in a network composed of 50,000 subnetworks?
How do you deploy SDN in a network composed of 50,000 subnetworks?

Well, you don’t ...
Instead, you aim at finding locations where deploying SDN can have the most impact
Instead, you aim at finding locations where deploying SDN can have the most impact on:

- Connect a large number of networks
- Carry a large amount of traffic
- Are opened to innovation
Internet eXchange Points (IXP) meet all the criteria

Deploy SDN in locations that

- connect a large number of networks
- carry a large amount of traffic
- are opened to innovation

AMS–IX

- 675 networks
- 3.2 Tb/s (peak)
- BGP Route Server
- Mobile peering
- Open peering...

https://www.ams-ix.net
A single deployment can have a large impact

Deploy SDN in locations that

- connect a large number of networks
- carry a large amount of traffic
- are opened to innovation

AMS-IX

675 networks
3.2 Tb/s (peak)
BGP Route Server
Mobile peering
Open peering...

https://www.ams-ix.net
\[ \text{SDX} = \text{SDN} + \text{IXP} \]
$SDX = SDN + IXP$

Augment the IXP data-plane with SDN capabilities
keeping default forwarding and routing behavior

Enable fine-grained inter-domain policies
bringing new features & simplifying operations
**SDX = SDN + IXP**

- **Augment** the IXP data-plane with SDN capabilities
  - keeping default forwarding and routing behavior

- **Enable** fine-grained inter-domain policies
  - bringing new features & simplifying operations

  … *with scalability and correctness in mind*

  - supporting large IXP load and resolving conflicts
SDX enables multiple stakeholders to implement policies and apps over a shared infrastructure.
Bringing SDN to the Internet, one exchange point at the time

1. Architecture
   programming model

2. Scalability
   control- & data-plane

3. Applications
   inter domain bonanza
Bringing SDN to the Internet, one exchange point at the time

1 Architecture
programming model

Scalability
control- & data-plane

Applications
inter domain bonanza
An IXP is a large layer-2 domain

Participant #1

Edge router

Participant #2

IXP Switching Fabric

Participant #3
An IXP is a large layer-2 domain where participant routers exchange routes using BGP.
To alleviate the need of establishing eBGP sessions, IXP often provides a Route Server (route multiplexer).
IP traffic is exchanged directly between participants
With respect to a traditional IXP,
With respect to a traditional IXP, SDX data-plane relies on SDN-capable devices.
With respect to a traditional IXP, SDX control-plane relies on a SDN controller
SDX participants express their forwarding policies in a high-level language, built on top of Pyretic (*)

(*) http://frenetic-lang.org/pyretic/
SDX policies are composed of a *pattern* and some *actions*

match ( Pattern ), then ( Actions )
Pattern selects packets based on any header fields,

**Pattern**

```
match (eth_type, vlan_id, srcmac, dstmac, srcip, dstip, protocol, dstport, srcport) && (tos, ||), then (Actions)
```
Pattern selects packets based on any header fields, while actions forward or modify the selected packets.
Each SDX participant writes her policies independently

Participant #2 policy

\texttt{match}(\text{dstport}=80), \texttt{fwd}(\#3)
\texttt{match}(\text{dstport}=22), \texttt{fwd}(\#1)
Each SDX participant writes her policies independently

Participant #2 policy

match(dstport=80), fwd(#3)
match(dstport=22), fwd(#1)

Participant #3 policy

match(srcip=0*), fwd(left)
match(srcip=1*), fwd(right)
... and transmit them to the SDX controller

Participant #2 policy

\texttt{match(dstport=80), fwd(#3)}
\texttt{match(dstport=22), fwd(#1)}

Participant #3 policy

\texttt{match(srcip=0*), fwd(left)}
\texttt{match(srcip=1*), fwd(right)}
The controller compiles all the policies into SDN forwarding rules.

**Participant #2 policy**

- match(dstport=80), fwd(#3)
- match(dstport=22), fwd(#1)

**Participant #3 policy**

- match(srcip=0*), fwd(left)
- match(srcip=1*), fwd(right)
SDX compilation stage implements *each* participant policy in the data-plane

Ensuring isolation

Resolving conflict

Considering BGP
SDX compilation stage implements *each* participant policy in the data-plane.

**Ensuring isolation**

Each participant controls one "virtual" switch connected to participants it can communicate with.

**Resolving conflict**

**Considering BGP**
SDX compilation stage implements *each* participant policy in the data-plane.

Ensuring isolation

Resolving conflict

Considering BGP

Policies are composed according to BGP business relationships.
SDX compilation stage implements 

*each* participant policy in the data-plane

Ensuring isolation

Resolving conflict

Considering BGP

Policies are augmented with BGP information
guarantee correctness and reachability
Bringing SDN to the Internet, one exchange point at the time

Architecture
programming model

2

Scalability
control- & data-plane

Applications
inter domain bonanza
The SDX platform faces scalability challenges in both the data- and in the control-plane.
data-plane

space

control-plane

time

512k prefixes, 500+ participants, potentially $10^9$ of forwarding rules

forwarding rules must be updated dynamically according to BGP
To scale, the SDX platform leverages existing infrastructure & domain-specific knowledge.
aggregate rules, on *existing* routers
SDX groups IP prefixes according to their behavior through the fabric

- policies are prefix-based
  just the way the Internet works

- forwarding actions are shared for a lot of prefixes
  \textit{e.g.}, all prefixes advertised by X
SDX groups IP prefixes according to their behavior through the fabric

- Policies are prefix-based just the way the Internet works
- Forwarding actions are shared for a lot of prefixes e.g., all prefixes advertised by X
- Group prefixes by equivalence class
SDX leverages edge routers to map packets to their equivalence class.

FIB size: $O(500k)$ IP entries

SDX
SDX considers edge routers’ FIB as the first stage of a multi-stage FIB.
Routers FIB match on the destination prefix and set a tag accordingly

Table #1

set a TAG based on IP prefix

Table #2

Edge router

SDX switch
SDX FIB matches on the tag

Table #1

set a TAG based on IP prefix

Table #2

match TAG

Edge router

SDX switch
SDX uses BGP NH as a provisioning interface and MAC addresses as tag in the data-plane.
SDX accommodates policies for 100+ participants, with less than 30k rules
data-plane

\textit{space}

control-plane

\textit{time}

leverage

\textit{policy structure}
SDX policies share key characteristics

Static
- disjointness

Dynamic
- locality
- burstiness
SDX policies share key characteristics

Static  disjointness

Dynamic  locality

burstiness

disjoint policies don’t need to be composed

significant gain as composition is costly
SDX policies share key characteristics

Static
- disjointness

Dynamic
- locality

Policy updates usually impact few prefixes
- 75% of the updates affect no more than 3 prefixes

burstiness
SDX policies share key characteristics

**Static**
- disjointness

**Dynamic**
- locality

**burstiness**
- policy updates are separated by large periods of inactivity
- In 75% of the case, updates are separated by 10s or more
These characteristics enable an efficient, 2-stage compilation algorithm

Stage 1  \textit{Fast}, non-optimal algorithm upon updates  
can install more forwarding rules than required

Stage 2  \textit{Slow}, but optimal algorithm in background  
regroup rules according to forwarding behavior
These characteristics enable an efficient, 2-stage compilation algorithm

- **Fast**, non-optimal algorithm upon updates
  - can install more forwarding rules than required

- **Slow**, but optimal algorithm in background
  - regroup rules according to forwarding behavior

- Time vs Space trade-off
In most cases, the SDX takes $<100$ ms to recompute the entire policy.
Bringing SDN to the Internet, one exchange point at the time

- **Architecture**
  - Programming model

- **Scalability**
  - Control- & data-plane

3. **Applications**
  - Inter domain bonanza
SDX enables a wide range of novel applications

<table>
<thead>
<tr>
<th>security</th>
<th>Prevent/block policy violation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prevent participants communication</td>
</tr>
<tr>
<td></td>
<td>Upstream blocking of DoS attacks</td>
</tr>
<tr>
<td>forwarding optimization</td>
<td>Middlebox traffic steering</td>
</tr>
<tr>
<td></td>
<td>Traffic offloading</td>
</tr>
<tr>
<td></td>
<td>Inbound Traffic Engineering</td>
</tr>
<tr>
<td></td>
<td>Fast convergence</td>
</tr>
<tr>
<td>peering</td>
<td>Application-specific peering</td>
</tr>
<tr>
<td>remote-control</td>
<td>Influence BGP path selection</td>
</tr>
<tr>
<td></td>
<td>Wide-area load balancing</td>
</tr>
</tbody>
</table>
SDX enables a wide range of novel applications

| security                  | Prevent/block policy violation |
|                         | Prevent participants communication |
|                         | **Upstream blocking of DoS attacks** |
| forwarding optimization  | Middlebox traffic steering |
|                         | Traffic offloading |
|                         | Inbound Traffic Engineering |
|                         | Fast convergence |
| peering                 | Application-specific peering |
| remote-control          | Influence BGP path selection |
|                         | Wide-area load balancing |
SDX can help mitigating DDoS attacks, closer to the source
AS1 is victim of a DDoS attack targeting its web server.
AS1 remotely installs *drop* policies in all SDXes.
AS1 remotely installs drop policies in all SDXes

AS1 policy

```text
match(srcip=*, dstip=10.0.01/32, dstport=80) >> drop()
```
SDX policies are targeted, hence other services stay reachable

AS1 policy

```
match(srcip=*, dstip=10.0.01/32, dstport=80) >> drop()
```

- single IP
- single service
SDX enables a wide range of novel applications

<table>
<thead>
<tr>
<th>Category</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>security</td>
<td>Prevent/block policy violation</td>
</tr>
<tr>
<td></td>
<td>Prevent participants communication</td>
</tr>
<tr>
<td></td>
<td>Upstream blocking of DoS attacks</td>
</tr>
<tr>
<td>forwarding optimization</td>
<td>Middlebox traffic steering</td>
</tr>
<tr>
<td></td>
<td>Traffic offloading</td>
</tr>
<tr>
<td></td>
<td><strong>Inbound Traffic Engineering</strong></td>
</tr>
<tr>
<td></td>
<td>Fast convergence</td>
</tr>
<tr>
<td>peering</td>
<td>Application-specific peering</td>
</tr>
<tr>
<td>remote-control</td>
<td>Influence BGP path selection</td>
</tr>
<tr>
<td></td>
<td>Wide-area load balancing</td>
</tr>
</tbody>
</table>
SDX can improve inbound traffic engineering
Given an IXP Physical Topology and a BGP topology,
Given an IXP Physical Topology and a BGP topology, Implement B’s inbound policies

B’s inbound policies

<table>
<thead>
<tr>
<th>to</th>
<th>from</th>
<th>receive on</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.0.1/24</td>
<td>A</td>
<td>left</td>
</tr>
<tr>
<td>192.0.2/24</td>
<td>C</td>
<td>right</td>
</tr>
<tr>
<td>192.0.2/24</td>
<td>ATT_IP</td>
<td>right</td>
</tr>
<tr>
<td>192.0.1/24</td>
<td>*</td>
<td>right</td>
</tr>
<tr>
<td>192.0.2/24</td>
<td>*</td>
<td>left</td>
</tr>
</tbody>
</table>
How do you that with BGP?

B’s inbound policies

<table>
<thead>
<tr>
<th>to</th>
<th>from</th>
<th>receive on</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.0.1/24</td>
<td>A</td>
<td>left</td>
</tr>
<tr>
<td>192.0.2/24</td>
<td>C</td>
<td>right</td>
</tr>
<tr>
<td>192.0.2/24</td>
<td>ATT_IP</td>
<td>right</td>
</tr>
<tr>
<td>192.0.1/24</td>
<td>*</td>
<td>right</td>
</tr>
<tr>
<td>192.0.2/24</td>
<td>*</td>
<td>left</td>
</tr>
</tbody>
</table>
It is hard

BGP provides few knobs to influence remote decisions

Implementing such a policy is configuration-intensive using AS-Path prepend, MED, community tagging, etc.
... and even impossible for some requirements

BGP policies **cannot** influence remote decisions based on source addresses

to from receive on
192.0.2.0/24 ATT_IP right
In any case, the outcome is **unpredictable**

Implementing such a policy is configuration-intensive using AS-Path prepend, MED, community tagging, etc.

There is *no guarantee* that remote parties will comply one can only “influence” remote decisions

Networks engineers have no choice but to “try and see” which makes it impossible to adapt to traffic pattern
With SDX, implement B’s inbound policy is easy

SDX policies give any participant *direct* control on its forwarding paths

<table>
<thead>
<tr>
<th>to</th>
<th>from</th>
<th>fwd</th>
<th>B’s SDX Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.0.1/24</td>
<td>A</td>
<td>left</td>
<td><code>match(dstip=192.0.1/24, srcmac=A), fwd(L)</code></td>
</tr>
<tr>
<td>192.0.2/24</td>
<td>B</td>
<td>right</td>
<td><code>match(dstip=192.0.2/24, srcmac=B), fwd(R)</code></td>
</tr>
<tr>
<td>192.0.2/24</td>
<td>ATT_IP</td>
<td>right</td>
<td><code>match(dstip=192.0.2/24, srcip=ATT), fwd(R)</code></td>
</tr>
<tr>
<td>192.0.1/24</td>
<td>*</td>
<td>right</td>
<td><code>match(dstip=192.0.1/24), fwd(R)</code></td>
</tr>
<tr>
<td>192.0.2/24</td>
<td>*</td>
<td>left</td>
<td><code>match(dstip=192.0.2/24), fwd(L)</code></td>
</tr>
</tbody>
</table>
SDX enables a wide range of novel applications

security
- Prevent/block policy violation
- Prevent participants communication
- Upstream blocking of DoS attacks

forwarding optimization
- Middlebox traffic steering
- Traffic offloading
- Inbound Traffic Engineering
- Fast convergence

peering
- Application-specific peering

remote-control
- Influence BGP path selection
- Wide-area load balancing
BGP is pretty slow to converge upon peering failure
Let’s consider a example with 2 networks, A and B, with B being the provider of A
Router B2 is a backup router, it may be used only upon B1’s failure
Both A1 and A2 prefer the routes received from B1 and install them in their FIB.

<table>
<thead>
<tr>
<th>prefix</th>
<th>NH</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>B1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>P500k</td>
<td>B1</td>
</tr>
</tbody>
</table>

forwarding table

500,000 BGP routes
Upon B1’s failure, A1 and A2 must update every single entry in their FIB (~500k entries)

Prefix  NH
P1      B1
...    ...
P500k   B1

forwarding table
Upon B1’s failure, A1 and A2 must update every single entry in their FIB (~500k entries)

Prefix  NH
P1  B2
...
...
P500k  B1

Forwarding table

Backup

500,000 BGP routes
Upon B1’s failure, A1 and A2 must update every single entry in their FIB (~500k entries)
On most routers, FIB updates are performed linearly, entry-by-entry, leading to slow BGP convergence.

\[
\text{convergence time} \quad \frac{500k \text{ entries} \times 150 \ \mu\text{secs}}{\text{entry}}
\]

average time to update one entry
On most routers, FIB updates are performed linearly, entry-by-entry, leading to *slow* BGP convergence.

\[
\text{convergence time} = \frac{500k \text{ entries} \times 150 \ \mu\text{secs}}{\text{entry}} = O(75) \text{ seconds}
\]

average time to update one entry
With SDX, **sub-second** peering convergence can be achieved with **any** router
When receiving multiple routes, the SDX controller pre-computes a backup NH for each prefix.
When receiving multiple routes, the SDX controller pre-computes a backup NH for each prefix.
Upon a peer failure, the SDX controller directly pushes next-hop rewrite rules.
match(srcmac:A1, dstmac:B1), rewrite(dstmac:B2), fwd(B2)
match(srcmac:A2, dstmac:B1), rewrite(dstmac:B2), fwd(B2)
All IP traffic **immediately** moves from B1 to B2, independently of the number of FIB updates.
SDX data-plane can enable sub-second, prefix-independent BGP convergence.

Convergence time = \( \# \text{edge entries} \times 150 \ \mu\text{secs} + 30\sim50 \ \text{ms} \) (per entry)

Controller communication time
SDX data-plane can enable sub-second, prefix-independent BGP convergence

\[
\text{convergence time} = \text{\# edge entries} \times 150 \ \mu\text{secs} + 30\sim50 \ \text{ms} \\
\text{entry} = \text{O}(30\sim50) \ \text{ms}
\]
SDN devices can boost the performance of traditional devices. Prototype under way!

Logical unit

- old router (Cisco 7200)
- cheap SDN switch
- high-end router (Cisco CRS 12000)
Bringing SDN to the Internet, one exchange point at the time

Architecture
programming model

Scalability
control- & data-plane

Applications
inter domain bonanza
SDX is a promising first step towards fixing Internet routing

Enable declarative, fine-grained inter-domain policies
many of which are not possible Today

Scale to hundreds of participants
both in the control- and in the data-plane

Running code (*) and deployment under way
important potential for impact

(*) https://github.com/sdn-ixp/sdx-platform
Bringing SDN to the Internet, one exchange point at the time

Laurent Vanbever
www.vanbever.eu

COS 561
November, 11 2014