Objective

Set out to reverse engineer SDN implementations and secure the entire thing.

Unable to find production implementations aside from intraAS ones in big data centers!

SDN: More Questions than Answers

* How to exchange SDN policies?
* Authorization and Authentication between Controllers?
* How do you securely peer applications between AS's?
* How is the level of access between multiple domains managed?
* How is flow rule arbitration done as applications produce complex interdependencies?
* and many more…

Motivation

* Bridge to SDN – will have hybrid BGP/SDN environments for years – SDN cannot and will not be "greenfield"
* OpenFlow interoperability across the WAN, between AS's. Most current implementations are intra-AS
* Eliminate Controller as Single Point of Failure
* Authenticate and validate resource exchange – implicit trust shouldn't be granted simply due to location
* No means to exchange SDN policies *vs. exposing raw infrastructure*
* What does it take to do peering between BGP and SDN?
* Just try something!

Our Goals

* Build a Multi-domain, distributed SDN architecture
* Better inbound path declaration
* Shared policy across sites
* No need for topology awareness, nor heavyweight L2 path signaling
* No low-level (OpenFlow) details exposed
* “My AS, My policy”
* Just drop rules that are in conflict – keep it simple!
* Similar functionality to existing BGP peering model
* Authorization and authentication scheme built into it from beginning
* Auto re-route based on predefined criteria
* Make it easily adoptable by operators

Current Network Security Challenges with BGP

* Too many humans involved in process – ie communicating new prefixes to other networks is manually intensive
* Based on Trust – security issues are largely social not technical
* Perception of "too much work" in instituting best practices
* Security best practices often ignored. Following these would rectify many security issues:
  * BCP38
  * RPKI
  * BGPSec
  * Relevant ingress/egress filtering
  * Not building policy from Internet routing policy DB

Security Challenges in SDN*

* Forged Traffic Flows
* Attacks on switches
* Attacks on control plane communications
* Attacks on controller vulnerabilities
* Lack of means to ensure trust between controller and apps

* Taken from paper "Towards Secure and Dependable Software-Defined Networks" , Kreutz et al

SDN Security Standards

* SDN gaining popularity within IETF, IEEE and ACM. New ACM conference "Symposium on SDN Research" at 2015 Open Networking Summit
* Most successful SDN deployments are intra-WAN: Google B4, Verizon and other service providers, so security is much less of a concern
* Open Networking Foundation (ONF) is working toward a security standard
* Security optional now in OpenFlow (TLS on control channel)
* AL2S has a shim layer (flowspace firewall) - policy engine indicates which vlan you can manipulate. some security is implemented for what they expose. Since very little is exposed, this is easier.

Things We Wanted to Avoid

* The need to know structure of peered network
* Manually calling out each interface
* Granting permissions for a Controller outside your AS to make network changes
* Heavyweight circuit signaling: NSI/Oscars
* Writing a lot of code

The SDX Concept

* SIGCOMM 2014 paper: "SDX: A Software Defined Internet Exchange", Feamster, Rexford etc. Mimics the concept of an IXP but with SDN – AS's connect and share policies (via a route server), which SDX combines into one. In this model it's all or nothing in terms of policy sharing
* In reality, ISPs reluctant to share policies with each other
* Constant competition between ISPs – their policies give them a competitive edge.
* Governance issue at SDX

Is this an SDX?

* IXP Architecture + SDN = SDX
* Why do you need a controller at an SDX? It buys you a single pane of glass for big network but why is this needed at an SDX?
* SDX= Layer 2 focus.
* Our design leverages existing protocols to better utilize Layer 3 instead

Other interesting SDN Projects

* SDX:
  * GaTech/SOX, Feamster
  * Starlight/Mambretti
  * NZ – City Link-NoviFlow
* Project Cardigan – Google New Zealand
* VeriFlow – layer between controller and devices that checks for changes in real time
* Fresco – incorporates FortNOX security enforcement kernel
* GENI
* SciPass

New Zealand: Project Cardigan

2012 - Citylink and REANNZ were considering SDNs as a future direction. Desired proof of concept.
SDN Controlled Fabric at an Internet Exchange.

2013 - Open Flow Controller + RouteFlow custom code = network fabric in production

Passes production traffic. Uses hierarchy of rules.

Flowspec

* Circa 2009
* Only available on some carrier-ish grade stuff.
* Not widely used, hardware implementations so-so.
* Generally used for anti-DDoS, blackhole, and FW uses
* Typically not open to customers to input rules safely.
* Flexibility of all tuning you can do with BGP.
* SDXs don't have ability to dictate low level communication between two parties.

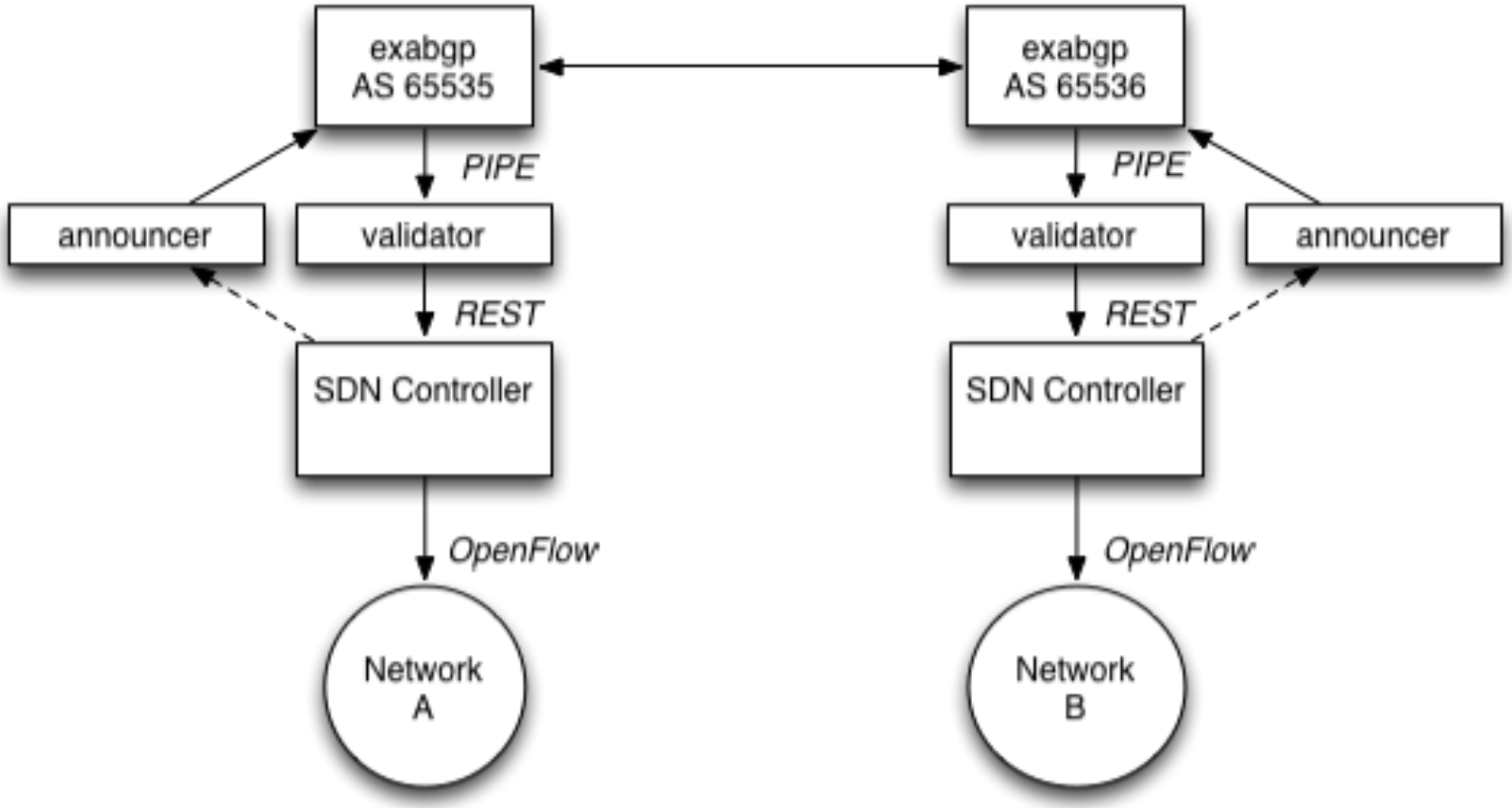
Flowspec is Key

- Contains most/all? matching fields we want to transmit flow policy between autonomous systems via BGP.
- Plus, we get all the proven BGP features: a formalized notion of "peering", route reflectors, confederations, tagging with communities!!!, etc.
- Already in some production use, particularly for real-time blackhole, however nearly only intra-AS and tightly coupled to hardware implementations. (Our motivation is inter-AS, and loosely coupled).

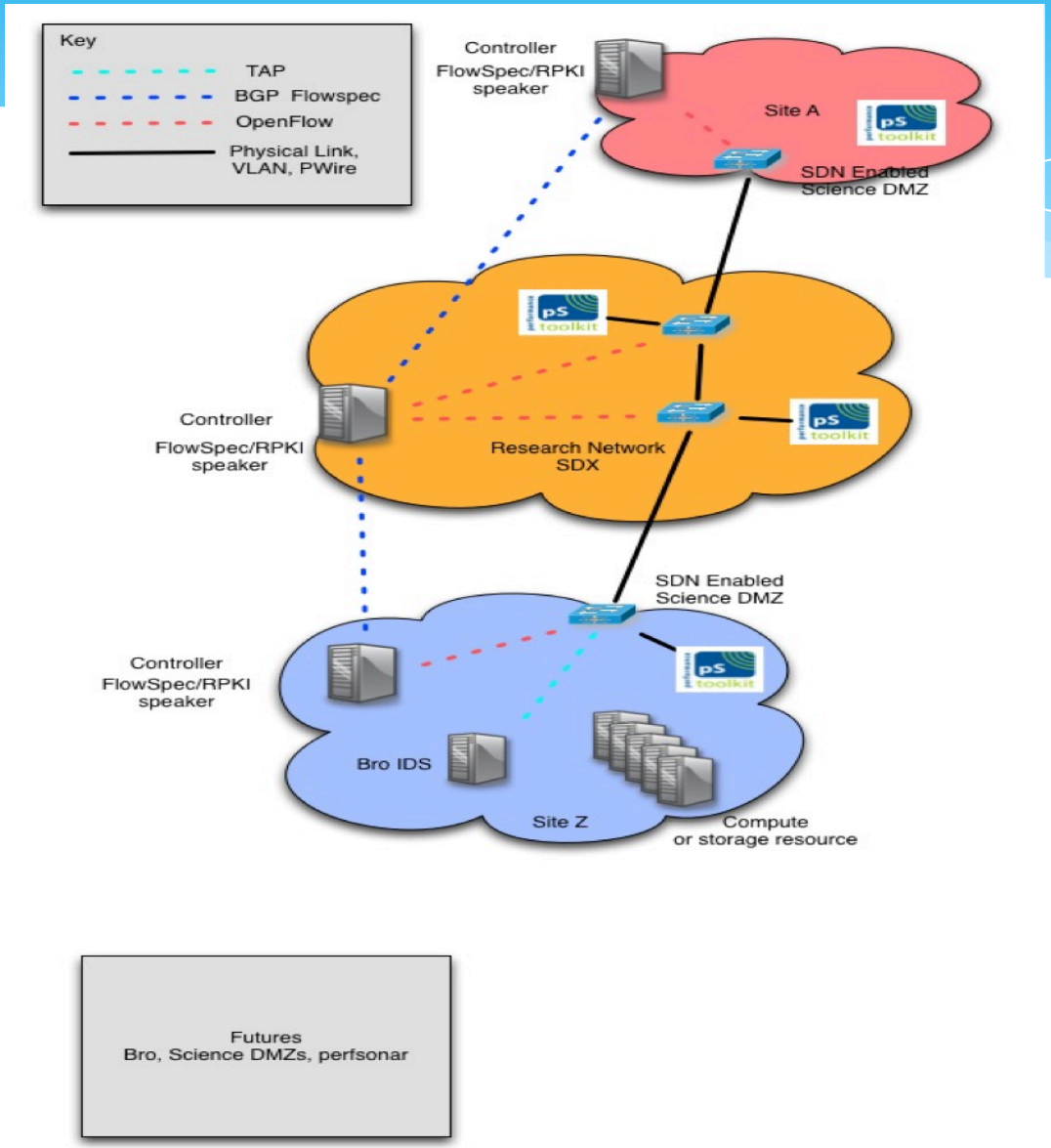
A flowspec rule

```
flow {
 route my-magic-thingy {
        match {
               source 2001:db8:48::/64;
               source-port >1024;
               destination 2001:db8:9000::/64;
               destination-port =22;
               protocol [ tcp ];
         } then {
                        community [65000:543];
                          }
             }
}
```

Logical Topology



Physical Topology



Implementation Example

* **Goal:** Decouple Inter-AS SDN policy from openflow/ hardware coupling.
* Add semantic meaning to flows with flowspec actions & BGP communities.
* Example use case: Site 'A' signals to 'B' "Hey, I have this ScienceDMZ prefix that shouldn't go through firewalls." 'B' decides this is valid, and translates the flowspec message into an OpenFlow rule specific to Site B". *No need for a L2 WAN circuit, L3VPN, etc.*

Announcer Function Example

* Something (SDN Controller infrastructure, or static policy) at site A determines it needs to put in a policy at site B to effect some action.
* This can be dynamic or just a static configuration.
* Signaled as BGP Flowspec policy rule from site A to the peer at site B.
* Site B now decides what to do with this, specific to its network, topology, and hardware capabilities. "My AS, My Policy".
* No further feedback given to site A (just like traditional BGP)

Validator Receiver example

* Site ‘B’ receives the flowspec policy from Site ‘A’.
* The Validator determines if this policy request should be installed or not:
  * - Does this match the unicast RIB?
  * - RPKI check (optional… there are issues)
  * - Max # of prefixes
  * - Rate limit updates from each peer.
  * - Appropriate action specified / community tag makes sense.
* Site B’s policy determines how to translate requests from ‘A’ into something site B’s SDN network can do.

New Features of this Design

* Use of Flowspec outside an AS, safely
* Integration of BGP and SDN into an interdomain mesh (not entirely new – see Cardigan)
* Can send traffic from an SDN site to a BGP site. Site A can be BGP Site B can be SDN
* Uses 12-tuple matching criteria which is inherent in OF
* Being able to skip a non SDN network in the middle – why go hop to hop anymore? Decouple policy conflict mechanism in validator from the controller
  * They can be upgraded independently
  * Organizations use different controllers

Technical Specs

* OpenFlow 1.3
* Switches:
* Controller: OpenDaylight.
  * Pros: Robust. Operator and user friendly.
  * Cons: Java.
* BGP Flowspec Tool: ExaBGP
* Code: DSS1.0 (Dale's Secret Sauce)
* Authentication: RPKI

Skip non SDN networks

* Just need to know peering end points, not every hop (it's multi-hop BGP)
* Use of BGP route reflectors in middle avoids having to configure a full mesh of peers at an exchange.
* Unlike a data center network, in which you need to know entire topology
* Control Plane doesn't care what's in the middle
* Use case: Regional network doesn't want to participate in SDN routing

RPKI

* May as well start with it from Day 1
* Today it's only for validation. Provides a foundation for transitive trust through an ISP
* Lessons learned by working with ARIN. Can't sign legacy R&E networks without a contract with ARIN
* Signing is the painful part.

SDN Security is Achievable

* Despite a lack of standards, the following can enhance SDN security:
  * RPKI - resource validation
  * Operator authentication - AAA
  * Event logging – Bro. Simple informational stuff we take for granted. Keep an informational log of what's happening. Transcript of what's happening.
  * Decoupling hardware from policy

Put in Science DMZ

* Peer more easily with other Science DMZs to optimize routing of "elephant flows"
* Split science flows from campus traffic. Route based on predefined criteria.
* Intercampus communications (LHC data among Tiers), test traffic, perfsonar node traffic all good candidates
* Good for HPC apps that dynamically moves large amounts of data

Integration with Bro IDS

* Bro has APIs.
  * PACF = Packet Acquisition and Control Framework
* Used for determining the correct path (via DPI) to do "fast paths" – looks for appropriate protocols – GridFTP, signal into next site to carve out path which is more optimal
* Site A is compromised, stop traffic to/from questionable resource(s).

Uses: R&E & Commodity Networking

* Improved Traffic Engineering for event driven instances:
  * Send large amounts of data to and from commodity networks - ie from R&E campuses to Google of for video traffic like the superbowl
  * Manage data movement from R&E/science onto AWS if burstable compute is needed, for example very large scale simulations. Use it for managing data movement in and out of AWS. Better connectivity in general is needed. Once it's in place, having reservations to those resources is important.
  * Apple iOS updates. WiscNet, I2 & other networks had to manually upgrade to 20Gb to Apple for updates
  * Regional Network to I2 Backbone Traffic Engineering. Avoid pile of VLAN spaghetti seen in regional network requests from members. No 3rd party needed.
* Adds to flexibility in Traffic Engineering even to non-R&E Networks

Benefits

* Redirect large science flows and route based on predefined criteria
* Takes advantage of Open Flow 1.3's N tuple matching and its ability to do a more fine grained protocol matching for traffic.
* Dynamic provisioning. More automated traffic engineering
* Improvement over L2 SDN because skip hop to hop.
* Mixes BGP - a well documented protocol - with innovative uses of a new and somewhat unknown protocol
* Solves BGP routing table leaks because built with best practices

Other things we considered

* OSCARS – usability is key. Dynamic paths are useful for optimizing flows
* PCE (Path Computation Element). RFC 4655.
* MPLS – Fairly complicated, requires more expensive hardware, adoption is in specific areas of networking whereas SDN can be more overarching and in all area

Lessons Learned

- SDN is still hype. Production usage is relatively small for niche uses.
- Set out to reverse engineer SDN implementations and secure the entire thing but unable to find production implementations aside from intraAS.
- Eliminate need to know all networks' topologies
- Build bigger tools from a set of smaller working parts. At least until we know what we're doing. "sysV init vs SystemD"
- Thought industry was further along than they are. We initially sought to take existing solutions & implement a proof of concept. Instead, 4 months later we decided to just try "something" (fling spaghetti at wall) and see what happens.

Roadmap/Future

- Integration with Globus
- Integration with XSEDE. XSEDE already uses I2's AL2S.
- Integration with HTCondor - Lark project can dynamically call for network rules to be inserted (and removed) per-job to control access.
- Use for traffic engineering in Science DMZs
- Use for burstable compute resource access

Questions?