# A Software-Defined Flexible Inter-Domain Interconnect using ONOS

Himal Kumar†, Craig Russell⋆, Vijay Sivaraman†, Sujata Banerjee∗
†UNSW, Australia, ⋆CSIRO Data61, Australia, ∗Hewlett-Packard Enterprise Labs, US
Emails: himal.kumar@unsw.edu.au, Craig.Russell@data61.csiro.au, vijay@unsw.edu.au, sujata.banerjee@hpe.com

*Abstract*—**This paper describes the architecture, use-case, and evaluation of a software-defined interconnect. Interconnects at Internet Exchange Points (IXPs) today comprise a layer-2 data-plane and BGP control-plane. We architect, implement, and deploy an SDN-based IXP using ONOS that provides equivalent functionality, while additionally supporting better ARP hygiene and enhanced traffic telemetry. Our system also provides private Internet peering, such as between an enterprise and a cloud provider. We demonstrate a novel use-case enabled by our system, wherein an enterprise can dynamically and transparently switch between public and private peering to access cloud services at appropriate quality and cost points. Evaluation of our solution with real traffic demonstrates that it enables enterprises to manage cloud connect costs flexibly without compromising quality, opening the door to innovative solutions not possible before.**

## I. Introduction

Interconnections between network domains are growing at a rapid rate: content providers (like Netflix, Google, and Facebook) have strong motivations to increase their direct connectivity with Internet Service Providers (ISPs) to reduce network bottlenecks and latencies to improve user experience. Simultaneously, enterprises are seeking direct connections with cloud providers (like Amazon and Microsoft) to have better access to their own IT infrastructure hosted in the cloud, as well as with other enterprises with whom they exchange substantial traffic. The growing need for inter-connects is fueling business for Internet Exchange operators (IXPs) and private-cloud operators who are experimenting with different service and pricing models suited to the public and private peering needs of their customers.

The interconnect architecture remains relatively simple: the data-plane can be a layer-1 circuit in the case of a private inter-connect between two domains, and a layer-2 fabric for a public peering between several domains. The control plane is de-facto BGP, either between the two peers (in the private case) or between each domain's border router and the route-server run by the IXP (in the public case). The natural separation of the data-plane and control-plane in the inter-connect architecture lends itself to the application of software defined networking – this is nicely illustrated in [1] [2] [3], which shows that an SDN-based IXP architecture (called SDX) can replicate current functionality and additionally offer new features (such as traffic engineering and security mitigation) that are infeasible today.

Prior work, such as SDX, has focused on public inter-connects. In this paper, we extend the architecture to include private inter-connects, which are becoming increasingly important for enterprises seeking dedicated connectivity to their cloud providers (e.g. AWS DirectConnect and Azure ExpressRoute), or to other enterprises with whom they may be exchanging large traffic volumes (e.g. DropBox and TurnItIn). In fact, it is becoming increasingly common for an enterprise to have a public peering at an IXP in order to get general Internet access, and one or more private peerings at that IXP to access cloud providers and/or other enterprises in a dedicated manner. The aim of this work is to develop a system (CASToR) that can handle both public and private inter-connects in a unified manner, built on top of a commercial-grade SDN platform (ONOS). Our system can match functionality of existing IXPs, and additionally enable new use-cases that allow an enterprise to manage their traffic seamlessly across the public and private inter-connections. Our specific contributions are:

- We architect and implement a system called CASToR (Connectivity As-a-Service to Top-of-Rack) that provisions public and private inter-connects between domains at an IXP. CASToR is built on ONOS, a commercial-grade SDN controller, and currently deployed on a research test-bed in Australia inter-connecting 10 organizations. CASToR implements IXP hygiene (dropping multicasts and containing broadcasts), while supporting improved telemetry of inter-connect traffic volumes.
- We demonstrate a use-case that allows the IXP to seamlessly toggle traffic from an enterprise to its cloud provider between their public and private inter-connects, based on policy. We also develop an economic model that the enterprise can use to adjust the trade-off between quality and cost for its cloud traffic.
- We evaluate our system using traffic from a high-speed traffic generator, to validate its latency and loss performance when toggling traffic between public and private inter-connects, and quantify the associated economic benefits for the enterprise.

The rest of this paper is organized as follows: §II presents relevant background work on IXPs, SDN, ONOS, and cloud-connects needed for this work. In §III we present our CASToR architecture and implementation, while the cloud-connect use-case is presented in §IV. Our system is evaluated in §V, and the paper is concluded in §VI.

## II. Related Work

The use of SDN at an IXP has been explored in [1], which proposes a framework called SDX that allows policies to be specified and enforced at the public exchange in conjunction with BGP. This framework is extended in iSDX [4] to scale to support a large number of flow-rules resulting from the compilation of network policies so they fit into the limited hardware resources of the exchange fabric. The work at TouIX [2] focuses on migration of traditional exchange points to an OpenFlow-based architecture, emphasizing the need for innovation in current IXP architectures.

The above frameworks have largely focused on public peering at IXPs. With a growing number of enterprises moving their IT infrastructure to the cloud, public cloud providers have started offering direct private connects, such as Amazon DirectConnect [5] and Azure ExpressRoute [6]; such private connections are a growth area for IXPs, e.g. Equinix's "cloud exchange" platform [7]. Simultaneously, IXPs such as IIX are also offering private inter-connects [8] for enterprise partners that exchange a significant volume of traffic with each other. Our framework allows public and private inter-connects to be managed in a unified manner, as will become evident in the next section.

## III. CASToR Architecture

Our CASToR application is a controller based architecture for interconnections at an exchange point or a data center. It enables fast and dynamic provisioning of peers using a simple User Interface and caters to both public and private peering arrangements. CASToR gives the IXP operator the flexibility and desired control to add/delete/change a peering or interconnection in real time without maintaining any static configuration files. A high level CASToR stack architecture is shown in Fig. 1. We describe the structure and key components of the CASToR application next.

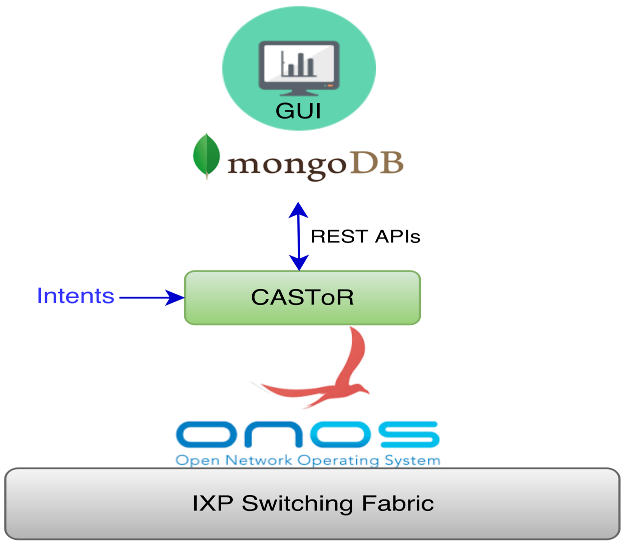


Fig. 1: CASToR Architecture

### A. The Platform: ONOS

CASToR is built as an application on top of the ONOS [9] controller. ONOS provides a scalable, high availability, industry grade controller platform to build intelligent applications thereby making it viable for large scale commercial networks such as an IXP. ONOS also provides a built in Intent-Framework [10] for installing low level flow rules into the network devices using abstract high level intents. CASToR exploits the Intent-Framework of ONOS to install flow rules into the IXP fabric using `MultiPointToSinglePoint`, `HostToHost` and `FlowObjective` intents which allows CASToR to be decoupled from the low level details and provide better APIs. CASToR uses OpenFlow as the south-bound protocol and the relevant switch drivers provided by ONOS to program the switches.

### B. Public Peering: Route Server

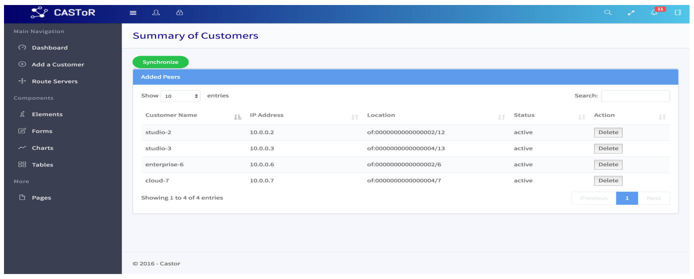
In our CASToR architecture, we use the Quagga [11] routing suite for the route servers at the exchange point. Every connecting peer runs a BGP session with the exchange route servers and advertise the prefixes reachable from them. The route server, Quagga then selects the best routes based on the BGP path selection parameters and metrics and advertise them back to all the peers. Route servers are only used for public peering where every peer receives the reachability information to every other peer connecting and peering at the IXP. In a public peering scenario all the route servers and the connecting peers are required to be configured to be in the same subnet for scalability.

### C. Private Peering

Our CASToR application also provides the ability to provision a private peering, which is essentially a cross connect between two peers who are willing to exchange traffic directly with each other. This can be motivated by the high volume of traffic being exchanged between these two entities, or a requirement of quality of service (e.g. low latency or high throughput) between these peers. A private peering typically entails establishing a dedicated BGP session running between the two peers. The IXP can provision a simple (layer-1 or layer-2) cross-connect to the peers so they can operate their own private (layer-3) BGP session. Our implementation has the capability to offload this task from the enterprise by proxying the BGP, in the form of a "virtual route-server" at the exchange, on their behalf. This virtual router-server is again a light Quagga instance which only runs the BGP control-plane, and no data-plane traffic ever passes through it. The detailed working of private peering is explained in §V.

### D. CASToR APIs and Interface

One of the objectives of CASToR is to enable the IXP operator to provision connectivity through the exchange via a simple web-interface. CASToR provides a rich REST API based north-bound interface that allows a web-based front-end GUI to be built, as well as an east-west interface for a customer to express intents.



Summary of Customers

Added Peers

| Customer Name | IP Address | Location | Status | Action |
|---|---|---|---|---|
| studio-2 | 10.0.0.2 | of:0000000000000002/12 | active | Delete |
| studio-3 | 10.0.0.3 | of:0000000000000004/13 | active | Delete |
| enterprise-6 | 10.0.0.6 | of:0000000000000002/6 | active | Delete |
| cloud-7 | 10.0.0.7 | of:0000000000000004/7 | active | Delete |

Showing 1 to 4 of 4 entries

© 2016 - Castor

Fig. 2: CASToR User Interface

*1) CASToR GUI:* We develop a proof-of-concept GUI for the IXP operator to provision the connectivity in the exchange fabric, as shown in Fig. 2 . A MongoDB backend is used for data persistence, and interacts with CASToR via REST APIs. To enable a peering, the operator enters the required information (Customer Name, IP address, Connecting Port, DPID of the switch) into the front-end interface, which then communicates and stores this information into the backend database. The backend synchronizes this information to CASToR to provision the required connectivity.

*2) CASToR Intent APIs:* CASToR exposes REST based APIs to the connecting peers to pass specific intents or policies desired by them. Peers pass simple JSON formatted data to CASToR to express an intent to achieve a desired state of the network pertaining to the performance of their connection at the IXP. An example of such an intent, which we illustrate in the next section, is to switch the traffic between the public and private inter-connects based on some preset performance criteria.

*3) ARP Hygiene:* ARP storms have always been a concern at the IXPs and its minimization is crucial [12]. Hundreds of peers may ARP for their next hops which may lead to significant congestion in the layer-2 exchange fabric leading to what is called ARP storms. In our SDN based architecture, the controller has the knowledge of the exact locations of the peers (DPID and Port number where peer is connected). CASToR uses this topology information provided by the ONOS controller to unicast the ARP traffic appropriately, preventing ARP floods. CASToR installs apposite flow rules into the switching fabric to match the ARP traffic and unicast it to the right switch/port. To achieve this, CASToR uses the `MultiPointToSinglePoint` ONOS intents where the single point is equivalent to the peer having the target protocol address of the ARP packet and multi-points would be all other remaining peers who can send ARP packets. The intent results in installation of flows into the fabric to match on the *Target Protocol Address* and *EthType ARP (0x0806)* and output it to the correct ports. These rules are best installed in a separate table on the switch if the hardware and drivers support it; at present our rules are installed in a single table to maintain generality.

## IV. Cloud-Connect Use case

We now illustrate how our CASToR platform supports Cloud-Connect in a cost-effective manner. An enterprise's cloud traffic typically has high temporal variation – this could be due to the enterprise's private data center load overflowing on to the public cloud provider during peak times (known as Cloud Bursting) or due to multiple traffic types that have different usage patterns. For example, our University's traffic to/from the Amazon cloud over the past 12 months is shown in Fig. 3 – the variability is accountable to both the nature of cloud bursting, and to the fact that student access to streaming video (e.g. Netflix) is sourced from AWS. Provisioning the private inter-connect to deal with peak load is expensive, while exchanging traffic over the public inter-connect is susceptible to quality fluctuations. We show how our platform enables a flexible peering arrangement that can help achieve the best of both worlds.

### A. *Flexible Hybrid Peering*

In the current peering model at an exchange point, an enterprise peers with a cloud provider either on public or private peering. There is a fixed cost associated in provisioning both types of peering charged by the IXP. Once a peer or customer decides whether to choose public or private, the provisioning is very much static and not changed unless required. Most of the IXPs charge a monthly fixed port fees for both types of peering in addition to one time costs like equipment and relocation costs, as explained in [13]. We propose in our architecture to use private and public peering together in a hybrid model to optimize cost and maximize performance based on volume-based dynamic pricing. An enterprise can have the flexibility to switch between private and public based on real-time traffic performance. This provides an enterprise the ability to trade-off their traffic between cost and performance and creates an opportunity for the IXP to run it as an extra value added service for economic gains.

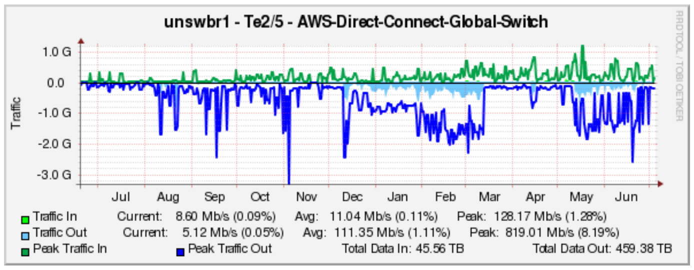


Fig. 3: AWS Traffic Loads of UNSW

An enterprise may want to switch its traffic or a part of it from public to private to gain better performance and low latency depending on the type of traffic. Since the public link is a shared link, one may not always get the desired performance on it. Switching to private peering comes at a higher cost but better performance which makes it inevitable to have control over switching in real time as per needed to save costs and maximize performance. Fig. 4 shows a high level architecture of the proposed hybrid model. An enterprise would usually have a single point of connection to the IXP fabric which bifurcates into two, public and private, to connect to the cloud provider. Dotted lines show that the enterprise is able to communicate with CASToR using the exposed APIs to pass intents.

**Public Cloud-Connect:** In Public Peering, an enterprise peers with the route servers at the exchange point using BGP. The enterprise runs a BGP session with the route servers and advertises its prefixes which are then further advertised and propagated to other peers which are peering with the IXP on public peering arrangement. In our architecture, public peering occurs exactly in the same way as it would at a traditional exchange point. A peer would get its next hop addresses and AS path from the route servers and then the data would be transmitted at layer-2.

**Private Cloud-Connect:** Private peering is a direct path between two peers like a cross-connect and provides a shortest high bandwidth path with only one next hop though at a higher cost. Private peering delivers better quality of services and traffic control but can also be more expensive. An enterprise would usually have a public peering at an IXP but can also peer privately with a cloud provider depending on business and traffic needs. There are two major challenges to support hybrid peering with traditional architecture: First, the enterprise will need to have a separate dedicated link for private peering on a different subnet to peer with the cloud provider privately and will also have to run an extra BGP session on it. Secondly, an enterprise does not have any visibility into the link states of other peers or cloud at the IXP to make switching decisions themselves. In our architecture, the IXP runs the dynamic private peering on behalf of an enterprise as a service to provide better performance. The enterprise only needs to have a single link to the IXP to have both types of peering. The IXP runs a virtual route server for the enterprise which runs a BGP session with the cloud on a separate private link and announce the prefixes for the enterprise to the cloud. The enterprise now has two BGP sessions with the cloud, one being run by the IXP which results in cloud receiving two next hops for the prefixes being advertised. These prefixes may or may not overlap in case of public and private depending upon what traffic needs to sent on which path. This is solely decided by the enterprise. In this way, the IXP provides the enterprise with the capability of dynamic hybrid peering as a service without exposing any information about network states of other peers.

## B. *Intent Based Switching*

We introduce Intent Based Switching [14] to toggle traffic dynamically between the public and private inter-connects, in order to realize the flexible hybrid inter-connect. The high-level intent passed by the enterprise to the IXP, in order to switch traffic from the public to the private inter-connect, could be as follows:

*Switch my cloud traffic from public to private if there is congestion on the public cloud link, with following parameters: Source* : A group of source prefixes for which traffic is to be toggled; example: {192.168.1.0/24, 192.168.2.0/24}. *Destination* : A cloud provider; example: {Amazon}. *Conditions* : Cloud public link utlization exceeds threshold $C_H$; example: {Condition: Destination Link Capacity, $C_H$=0.7}. *Actions* : Toggle; example: {to Private inter-connect}. A similar intent for reverse action, namely moving traffic from the private back to the public inter-connect, can be defined to take effect whenever the load of the Cloud provider's public link falls below threshold $C_L$, of say 0.4.

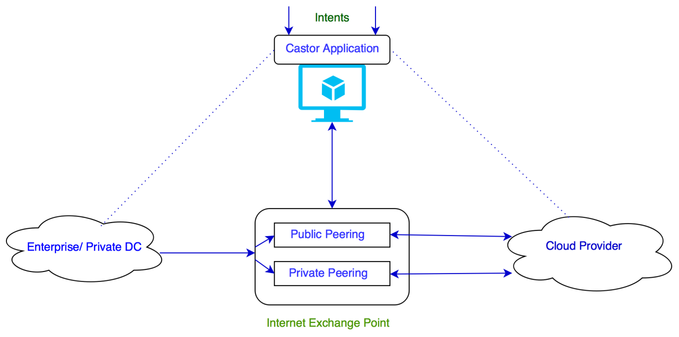


Fig. 4: Proposed Peering Architecture

Our CASToR application performs the switching by computing a binary parameter Z that takes value 1 for private peering and 0 for public peering for this particular enterprise-cloud relationship. Z is calculated by the application iteratively every minute as per below, where C is the current utilization of the public link to the cloud provider.

$$Z_{i+1} = \lceil C - C_H \rceil + Z_i * \lceil C - C_L \rceil \tag{1}$$

Whenever $Z_{i+1}$ differs from $Z_i$, the traffic is toggled between the public and private inter-connects. The implementation details of how this is achieved is described in §V.

## C. *Economics and Costing Model*

Both types of peering have two cost components - fixed and variable. The variable cost of public peering is expressed as $P_{pub} = A \sum_{i=1}^{n} X_i(1 - Z_i)$ and for private peering as $P_{pri} = B \sum_{i=1}^{n} X_i * Z_i$, where:

- A : This is cost of peering per unit volume on a public link.
- B : Cost of peering per unit volume on a private link.
- X : The volume of traffic from that enterprise to the various peers on either public or private inter-connect.
- Z : Switching binary variable: 0 for public peering and 1 for private.

This variable cost is calculated on a minute or on an hour basis. The fixed component includes the cost of provisioning a physical private-connect and peering port fees and is charged monthly by the IXP.

We expect $B > A$, namely sending a byte of traffic on the private inter-connect costs more than on the public inter-connect. The simplified total variable cost for an enterprise to connect to a single cloud provider is therefore:

$$P_{tot} = A * X * (1 - Z) + B * X * Z \tag{2}$$

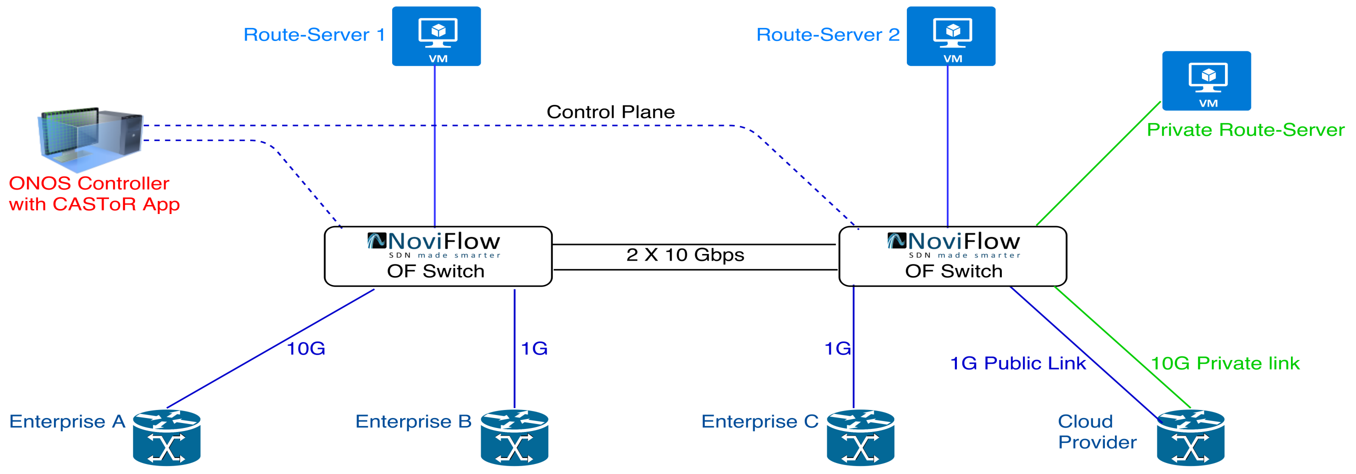


Fig. 5: Experimental Setup

where X denotes the volume of traffic the enterprise sends to a single cloud on private or public inter-connect. Our application calculates the value of Z, and the volumes of traffic to calculate the total cost and provide it to the enterprise as a feedback so that they can make cost optimization decisions by passing intents.

## V. Prototype Evaluation

Fig. 5 shows our experimental network comprising an SDN enabled exchange point using the CASToR application. It consists of two OpenFlow switches (NoviFlow NS1132s) [15] with two 10Gb/s inter-switch links acting as the exchange's layer-2 fabric. We assume that the fabric has infinite capacity with respect to bandwidth and is therefore never a bottleneck. Fig. 5 also shows two public route servers (Route-Server 1 and 2) used for BGP sessions with the Peers, as is typical in most current IXPs, as well as a private route server and an ONOS controller that runs the CASToR application. In the experiments we used four peers connecting to the exchange point, three enterprises (A, B and C) and one Cloud Provider that connect via border routers as shown. The border routers used for the experiments were two Cisco 3800 series routers, another NoviFlow NS1132 switch controlled by the Vandervecken virtual routing software [16] and a Pica8 P3295 [17] switch also controlled by Vandervecken. We generated traffic for the experiments using a Spirent TestCenter system with separate physical interfaces connected to each border router. The route servers and the ONOS controller were all Ubuntu 14.04 virtual machines hosted in an OpenStack cloud environment. Only Enterprise A had a private peering enabled with the Cloud Provider shown by green lines. The IXP hosts a private route server on behalf of enterprise A to run a BGP session with the cloud on a separate interface and IP subnet. The three enterprise border routers (A, B and C) only peer with the two IXP public route servers. Dotted lines show the control plane connectivity between ONOS/CASToR and the IXP switch fabric while solid lines show the data plane connectivity.

We focus the following discussion on Enterprise A and the Cloud Provider and consider the others as general peers present at the IXP (which in general could number in the tens if not hundreds for a large IXP). For simplicity, we refer to Enterprise A and the Cloud Provider as just A and Cloud respectively. A has a single 10Gb/s physical link with the IXP fabric but two different data paths to the Cloud - private and (shared) public as shown. The Cloud has a 1Gb/s public link and a 10Gb/s private link with the IXP fabric. In our experiment, we purposely choose 1Gb/s public link to be able to congest it easily.

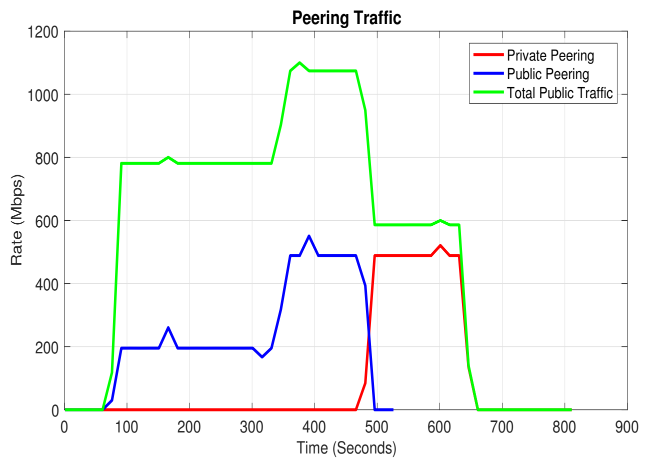


Fig. 6: Measured Peering Traffic

Each enterprise maintains BGP sessions with the two public route-servers and receives prefixes of the Cloud with the next hop addresses. The next hop to reach the Cloud on the public link is via the blue interface of the Cloud border router. Initially, all three enterprises (A, B and C) send traffic to the Cloud via the cheaper public link. Fig. 6 shows the corresponding traffic rate. The green curve shows the total public traffic going to the cloud, while the blue and red curves show the traffic from A to the Cloud via the public and private links respectively. Table 1 shows the number of dropped frames in different scenarios and the traffic rates of A to the Cloud. Initially A transmits 200 Mbps and the total public rate is close to 800 Mbps with no packet drops. At approximately 300s, A bursts to 500 Mbps and begins to experience packet drops as the total public rate exceeds the link capacity of 1 Gbps resulting in congestion on the public link. We can see from the table that A experiences an 11 percent loss of frames during this time. At this moment, A then passes an Intent to CASToR to change its traffic to the private link if the public link capacity exceeds 70 percent and back to public when it falls to 40 percent. CASToR then triggers the change to shift the traffic to private peering which happens in two stages.

| Average Measure of Dropped Frames | | |
|---|---|---|
| Type of Peering | Traffic Rate of A | Dropped Frames Count (%) |
| Public - No Congestion | 200 | 0 |
| Public - Congestion | 500 | 1052582 (11%) |
| Private | 500 | 0 |
| Public-Private transit | Any | 1-3 |

TABLE I: Measure of Dropped Frames

First, when a packet from A reaches the exchange fabric, CASToR instructs the OF switch to rewrite the source MAC to that of the private route server and the destination MAC to be that of the green interface of the Cloud's border router. It then instructs the OF switches in the fabric to output the traffic via the respective ports in order to change the path. It essentially overrides the default BGP behavior without requiring any action from A (or indeed without A's border router even being aware of the path change). A still thinks that its traffic is traversing the public link but in reality it is being switched to the private link by the IXP due to the intent. All the header rewrites are done by the OF switches in the data plane and no traffic goes to the private route server.

Second, the reverse path of the traffic from the Cloud to A is handled in a different manner. At this point the Cloud will continue to send reverse traffic on the public link since it has received no information to the contrary. This can be changed by using the BGP path attribute MED (Multi-Exit Discriminator). When a path switch is required, CASToR runs an automatic script which changes the MED value of the BGP announcement from the private route server to a lower value compared to that from the public route servers. This results in the Cloud's preferred path to be the private link for those particular prefixes. As a result of this change, when a packet from the Cloud arrives at the IXP fabric on the private link, it is destined to the MAC address of the private route server. Note that the AS number of the private route server is same as A's, making the AS path the same over both the private and public links. CASToR installs rules in the OF switch to rewrite the destination MAC to the actual MAC address of A and forwards it to the correct port to reach A. We can see in the figure that at this stage the traffic shifts to the private link shown by the red curve and the total amount of public traffic reduces. Both stages happen quickly without dropping any packets in transit. We purposely ran the experiment in the congested state for some time to show the congestion and packets being dropped. However, in a real scenario A would pass the intent and specify the switching parameters in advance and CASToR would enable an almost instant switchover thereby preventing congestion on the public link. We experienced a very small number of dropped frames (1 to 3) when the switchover occurred. The whole switchover process happens in near real time in an automated fashion without any intervention from enterprise A or the Cloud. This switchover can be done for any group of source or destination prefixes based on the requirements of the enterprise. For example, an enterprise may wish to keep latency sensitive traffic always on the private link or it may keep the low priority traffic on the public link to reduce transit costs. Switching to private peering can therefore be used to provide better performance and potentially a lower latency path.

## VI. Conclusions

As interconnections between enterprises and cloud providers are proliferating, there is a need to have a flexible architecture which can cater to the dynamic traffic changes and deliver better quality of service and customer satisfaction. This paper is an attempt to add flexibility and innovation to the IXP ecosystem to support dynamic hybrid peering which can be provisioned using a web GUI. We developed a prototype and implemented our architecture on a carrier-grade platform and quantified the associated economic and quality of service benefits. We validated our design and implementation using experiments on real systems and hardware and deployed our system on an Australia wide SDN testbed.

## References

[1] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "Sdx: A software defined internet exchange," *ACM SIGCOMM*, 2014.
[2] R. Lapeyrade, M. Bruyere, and P. Owezarski, "Openflow-based migration and management of the touix ixp," *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016.
[3] J. Stringer, D. Pemberton, Q. Fu, C. Lorier, R. Nelson, J. Bailey, C. Correa, and E. Rothenberg, "Cardigan: Sdn distributed routing fabric going live at an internet exchange," *Computers and Communication (ISCC)*, 2014.
[4] A. Gupta et al, "An industrial-scale software defined internet exchange point," *NSDI*, 2016.
[5] "Amazon direct connect," https://aws.amazon.com/directconnect/.
[6] "Microsoft azure expressroute," https://azure.microsoft.com/en-us/documentation/articles/expressroute-circuit-peerings/.
[7] "Equinix cloud exchange," http://www.equinix.com.au/resources/data-sheets/equinix-cloud-exchange/.
[8] "Iix console," http://www.iix.net/.
[9] "Open network operating system," http://onosproject.org/.
[10] "Onos intent framework," https://wiki.onosproject.org/display/ONOS/Intent+Framework.
[11] "Quagga," http://www.nongnu.org/quagga/.
[12] V. Boteanu, H. Bagheri, and M. Pels, "Minimizing arp traffic in the ams-ix switching platform using openflow."
[13] W. B. Norton, *Internet Peering Playbook*.
[14] "Onf blog, intent: What, and not how," https://www.opennetworking.org/?p=1633&option=com_wordpress&Itemid=155.
[15] "Noviflow," http://noviflow.com.
[16] "Vandervecken," https://github.com/routeflow/routeflow/tree/vandervecken.
[17] "Pica8, white box sdn," http://www.pica8.com/.