Contents

1 Introduction and Background .............................. 2
  1.1 Software Defined Networking ......................... 2
  1.2 Setting ............................................. 2
  1.3 Existing CMS Data Analysis Software Tools .......... 3
    1.3.1 PhEDEx ......................................... 3
    1.3.2 HTCondor ......................................... 3
  1.4 CMS Network Topology and Physical Structure .......... 4

2 Problems .............................................. 5
  2.1 Scalability: Cross-product Explosion and Full Instantiation 6
  2.2 Resource Management ................................ 7

3 Solutions .............................................. 7
  3.1 Scalability ........................................... 7
    3.1.1 Publish-Subscribe Model ......................... 7
    3.1.2 Multi-Dimensional Packet Classification .......... 8
  3.2 Resource Management ................................ 9
    3.2.1 Definitions ...................................... 9
    3.2.2 Constraints ..................................... 10
    3.2.3 Redundancy Testing .............................. 11
    3.2.4 Secure Two-Party Scalar Product Protocol ..... 11
    3.2.5 Resource Management Algorithm .................. 11

4 Implementation ...................................... 11
  4.1 SFP and ESnet ...................................... 12
  4.2 Unicorn and Resource State Abstraction Demonstration 13
    4.2.1 Demo Topology .................................. 13
    4.2.2 Demo Results ................................... 13

5 Next Steps ........................................... 14
  5.1 Convergence ........................................ 14
  5.2 Routing Table Implementation ........................ 15
  5.3 Merging Pipelines .................................. 15

References ............................................. 16
Abstract

Software-defined networking (SDN) separates the control plane and data plane of a network, offering logically centralized and much more flexible network-control programming. SDN continues to grow in use and importance in many networking applications, yet existing Internet protocols are not flexible to benefit from many of its advantages.

The current de facto inter-domain routing protocol on the Internet, Border Gateway Protocol (“BGP”), is not sufficiently flexible for use in routing among SDN networks; to use BGP in such a setting results in efficiency, scalability, and stability concerns.

Moving beyond BGP’s traditional push architecture, we (1) design and outline a novel pub-sub protocol based on defining network information spaces – we call this the SDN Federation Protocol (SFP, for short) – that overcomes many of BGP’s SDN shortcomings, and (2) discuss the design choices and data structures involved in a SFP implementation, specifically how advertising of routing information and resource abstractions is done.

1 Introduction and Background

1.1 Software Defined Networking

Software Defined Networking refers to the logical and physical separation of the “control plane” and the “data plane” in a networking domain. This separation makes a network, originally a relatively static structure, much more adaptable, dynamic, and able to be more easily centrally managed [12].

SDN is becoming more prevalent in individual networks, and so a logical next step is to consider how SDN networks may interconnect. In a setting characterized by many resources being shared across domains, it is advantageous to connect these domains. This paper will refer to the interconnection of independent SDN networks as a federation of SDN networks or as a SDN federation.

1.2 Setting

One such setting marked by the sharing of resource across domains is the Compact Muon Solenoid (CMS) experiment at CERN, the European Organization for Nuclear Research. We will present SFP in this context to highlight the relevance and applicability of a new, innovative interdomain protocol in a scientific setting.

The CMS experiment, part of the Large Hadron Collider project at CERN, is one of the largest scientific collaborations in the world. Four thousand researchers from over
40 countries are working together to seek a deeper understanding of particle physics by analyzing enormous quantities of data generated by collisions in the LHC [17]. A major challenge across the federation of participating institutions is to share and to analyze relevant data in a geographically-distributed, multi-domain setting.

CMS currently uses a variety of technologies to facilitate the sharing and analysis of particle collision results.

1.3 Existing CMS Data Analysis Software Tools

1.3.1 PhEDEx

The Physics Experiment Data Export is the CMS tool for managing large scale data transfers among sites involved in the project for geographically-distributed computation [18].

1.3.2 HTCondor

HTCondor, developed by the University of Wisconsin, Madison is the scheduler and workload management system that CMS uses to orchestrate jobs across the sites located around the world. HTCondor provides scheduling, prioritization, resource monitoring, and resource management [19, 20].
CMS Network Topology and Physical Structure

CMS uses the Worldwide LHC Computing Grid (WLCG) to distribute data to storage and computation nodes located worldwide. The WLCG consists of 4 “tiers,” each of which provides a specific service to the system.

At the root, Tier 0, are the CERN Data Centre in Geneva, Switzerland and the Wigner Data Centre in Budapest, Hungary. Raw data is produced at the LHC in Geneva, stored in the two T0 sites, shared with Tier 1 sites, and reprocessed at the T0 sites when the LHC is not running [14, 15].

Tier 1 consists of 13 computing sites, each of which stores raw and reprocessed LHC data, performs large-scale data reprocessing, and shares the data with Tier 2 sites. Seven of the Tier 1 sites are involved in the CMS experiments. Although the initial design of WLCG coupled the processing and archiving functionality of the Tier 1 sites, a 2014 redesign decoupled these two purposes, and this opened up the possibility of taking advantage of free CPU at the Tier 1 level, essentially letting Tier 1 become a shared resource pool [14, 1].

Tier 2 sites are universities and other research institutions that provide computation for specific tasks. The WLCG has about 160 Tier 2 sites worldwide [14]. Between 2013 and 2015, the LHC took a hiatus to refurbish almost all aspects of the system [5]. During this time, Tier 2 structures, which were formerly organized in a hierarchical structure, became a mesh, with many connections among T2 sites as well as to T1 [10].
Tier 3 networks are “informal” connections, and they can be individual machines or computing clusters of an individual scientist [14].

Two dedicated, private, and complementary networks have been built to connect the various tiers together. The older LHC Optical Private Network (LHCOPN) is a dedicated, high-bandwidth network connecting Tier 0 and Tier 1 at 10 Gbps. LHCOPN uses layer 3 functionality built by Tier 0 and Tier 1, while layer 2 support comes from a collaboration of R&E networks. The majority of Tier 1 sites have multiple links to the central Tier 0 site. Most critically, it is worthwhile to note that LHCOPN uses BGP routing [10].

More recently, CERN developed the LHC Open Network Environment (LHCONE) to provide “any-to-any” (i.e., mesh) connection between Tier 1 and Tier 2 sites for more flexibility and dynamic storage usage. LHCONE is a collaboration between a multitude of research and education (R&E) networks, including ESnet, GÉANT, Internet2, and others: that is, many stakeholders are involved, from the Tier X sites, to the R&E networks, to CERN [10].

2 Problems

Traditional interdomain protocols are insufficient for use in the case of a federation of SDN networks, like the federation of institutions participating in the CMS experiment. Consider the problems associated with applying BGP, the current *de facto* interdomain routing protocol, in the context of SDN federation [9].
2.1 Scalability: Cross-product Explosion and Full Instantiation

A significant advantage of SDN is that it makes possible logically centralized, more flexible network-control programming. For instance, the OpenFlow communication protocol that gives access to the forwarding plane of a switch or router uses a 12-tuple of header information. As a result, SDN switches and routers can make decisions based on a large number of decision dimensions, including source and destination IP addresses and ports, as well as protocol and VLAN information. Compare this flexibility to BGP, which uses a single dimension (destination IP address) when making routing decisions [13].

The naïve extension of BGP to include multidimensional decision-making – that is, to condense the 12 fields used in OpenFlow matching into a single dimension for use in BGP – brings about a cross product explosion and scalability obstacles. By way of illustration, note that to route $N$ destinations based on even the TCP/IP 5-tuple of protocol and destination and source ports and IP addresses would require up to $N \times M \times P \times P \times A \times A$ entries in the routing table, where there are $M$ protocols, $P$ ports, and $A$ addresses.

Moreover, BGP is a fully-instantiated information exchange protocol: its routing information base must have an entry for each possible destination, and each autonomous system advertises the best route for every destination to its neighbors. With the larger decision space used in SDN (supra), it will not be efficient or feasible to share these tables with each neighboring AS.


2.2 Resource Management

SDN also offers the benefit of managing resource allocation (e.g., bandwidth) of non-independent flows, whereas vanilla BGP is blind to resource usage and offers domains no means of querying or responding to bandwidth usage across links for specific flows. Multi-flow resource querying is even more complex, as it must account for sharing links, flow concurrency, and internal sensitive information preservation, which is not considered in the BGP model [9].

[11] examines the data analytics trace from the CMS experiment over a 7-day period. They show that resources in such a multi-domain and geographically distributed setting are unbalanced across domains. Yet, extending existing resource management theory for single-domain clusters to the multi-domain CMS setting compromises the privacy of the member institutions and requires too much overhead to maintain an accurate resource availability graph.

3 Solutions

3.1 Scalability

There exist efficient data structures for performing longest prefix matching across multiple dimensions in software, however, some of these are unable to scale beyond two-dimensional rules. We present an extensible design using pipelines for SFP.

3.1.1 Publish-Subscribe Model

The core innovation of the SFP protocol is its novel pub-sub method for exchanging routing information among autonomous systems. SFP is an east-west bound protocol, that is, it implements the communication interface between peering networks [9].

For the rest of this section, suppose that we have two peering networks $A$ and $B$. Network $B$ can respond to queries from network $A$ about (1) how its control program $P_B$ will handle each packet that $A$ is interested in (we say these form a packet space), and (2) metrics (e.g., bandwidth) associated with a set of flows, a point in the flowset space. So, we say that the east-west bound view consists of these two predefined spaces, the packet space and the flowset space.

These queries take the form of subscriptions. Network $A$ sends subscription interests of the form $(\text{subspace}, \text{metrics})$ to $B$; $B$ can reply with the metrics associated with subspace. Network $A$ is not limited in the number of subscription requests that it can send: for instance, $A$ can have a subscription for a packet space to obtain
reachability information for a range of IP destination addresses, and $A$ can also have a subscription for the resource constraints associated with a given flowset.

Let us focus on the packet space protocol. To compactly instantiate a program $P$ over a packet space $Q$, we encode $P$’s behavior over subsets of the packet space as a pipeline of table-formatted routing information bases. Tables are easy to query and transmit efficiently. Note that any program instruction $I$ can be represented as a flow table whose fields match on the arguments of $I$ and whose actions correspond to $I$’s behavior on a given set of arguments. We merge flow tables based on shared header fields to further compress the program.

Selecting queried rules: As described above, when a network controller receives a SFP subscription query $Q$, it first decides which subset $Q$ it wants to respond to, then it queries its RIB to select the corresponding rules. The next section presents a data structure to allow such multifield queries to occur.

3.1.2 Multi-Dimensional Packet Classification

Existing data structures for longest prefix matching are not fully applicable to the use case necessary in SFP (see [7] for an overview of existing packet classification algorithms). Many of the existing ways of classifying packets are limited in the number of dimensions (e.g., the grid-of-tries structure can match on at most 2 fields).

We present two approaches. The first sacrifices some of the expressive power of multi-dimensional rules for fast lookup speed. The second preserves the full expressive capability of multi-dimensional rules.

Parallel Voting. Suppose we have $N$ rules and $k$ fields. With this technique, we search each dimension $i \leq k$ independently and in parallel for the best prefix or range match in that dimension. With the inclusion of a catch-all rule, each search returns exactly one rule. Each of the $k$ results votes on the action to take; plurality wins, and that is the action returned. Because each dimension is treated independently, updates to the rules are fast and easy.

There are two main issues with the Parallel Voting procedure that render it unfeasible for use in SFP. First, and most significantly, in forcing dimensions to be searched independently, we lose the expressive power of having rules of the form $(F_1 = i \text{ AND } F_2 = j) \mapsto \text{action}$, one of the main advantages of including multi-dimensional rules in the first place. Second, it returns a single action, not an entire rule (because rules are not guaranteed to be corresponding). While this may be sufficient for use in a local forwarding table, in SFP we want to share pipelines of rules with adjacent autonomous systems. This shortcoming could be overcome by constructing a new pipeline for the query response, however, based on either the results of the parallel search or the voting step.
Despite these downsides, it’s a fast procedure. We can implement “lookup tables” for each field as a radix trie, and so lookup and prefix matching both take $O(L_i)$, where $L_i$ is the length of the longest rule in field $i$. (In practice, this is faster than a typical trie in the case where many entries share long prefixes, because all comparisons take constant time.) Voting can take place in time linear in the number of dimensions in the pipeline, $O(k)$. Altogether, then, we can return a rule in $O(L + k)$, where $L = \max_{0<i\leq k, i \in \mathbb{Z}} (L_i)$.

Iterative Subtable Lookup. Now we present a different method that does not make the independence assumption of the different fields involved in a rule. We represent the pipeline as a sequence of single-dimension search problems arranged in a hierarchical structure. Again, we use a tree-based structure for lookup in a single-dimension, which makes possible significant flexibility in searching across varying and specific prefix lengths. We begin by matching on the first dimension, and the resulting entry contains (a) pointer(s) to a table of rules with $k-1$ fields that (implicitly) have a value in Field 1 matching the initial query. We then repeat the process recursively.

Using the same notation as above, this lookup procedure takes time $O(Lk)$. Without an efficient representation of these tables in space with clever use of pointers, however, this approach could suffer from an explosion of space usage across the $k$ fields.

3.2 Resource Management

We now present a method for representing a flowset’s resource usage as a system of linear inequalities. When domain $A$ subscribes to a flowset space from $B$ with a metric of “available bandwidth,” $B$ will reply with a system of linear inequalities describing the feasible region of resource allocation for that particular flowset.

For maximal efficiency, we present a system for eliminating redundant linear inequalities; this system preserves the privacy of member institutions by revealing only the minimal resource abstraction for each domain.

3.2.1 Definitions

We need to define some notation to explain the details of the minimal resource state abstraction system. For a given flowset $F$, let $\Pi_d(F)$ be the resource state abstraction associated with flowset $F$ passing through domain $d$. Then, the minimal, equivalent resource state abstraction for a flowset $F$ is

$$\bigcup_{d \in D} \Pi_d(F) = \Pi_D(F).$$
Note the feasible region that $\Pi_d(F)$ represents may be expressed as $\bigcup_{d \in D} \Pi_d(F)$, where $\Pi_d(F) \subseteq \Pi_d(F)$, $\forall d \in D$ and $\Pi_d(F) \cap \Pi_{d'}(F) = \emptyset$, $\forall d, d' \in D$. We suppose a semi-honest security model, assuming that each domain follows the protocol but is curious. Our goal is to find, for each domain $d$, the minimal resource abstraction component $\Pi_d(F)$, that no other domain knows the content of $\Pi_d(F)$ or $\Pi_d'(F)$ and the orchestrating application does not know the content in $\Pi_d(F) - \Pi_d'(F)$.

**Proposition 1.** Given a set of linear constraints $\Pi$ and a linear constraint $c$, $c$ is redundant to $\Pi$ if and only if (1) all the vertices of the convex polyhedron defined by $\Pi$ are in the halfspace defined by $c$, or (2) all the vertices of the convex polyhedron defined by $\Pi$ are in the complementary halfspace of that defined by $c$.

### 3.2.2 Constraints

Suppose we have a set of flows $F$ and a domain $d$, and suppose that $n_d$ flows enter $d$. Then the domain resource constraints $\Pi_d(F)$ are composed of two parts. First, we have the constraints on abstract network elements, e.g., links. Second, we have the non-negative constraints of the flow rates.

Let’s first examine the abstract network element constraints. If we have $m_d$ such constraints in domain $d$, then we can represent the constraints as

$$Cr \leq b,$$

where $C \in \mathbb{R}^{m_d \times n_d}$, $r \in \mathbb{R}^{n_d}$, $b \in \mathbb{R}^{m_d}$. Row $i$ in $C$ represents the flow coefficients of each of the $n_d$ flows passing through abstract network element $i$. The vector $r$ is the vector of flow rates, and $b$ is the vector of available bandwidths of each abstract network element.

Now, let us consider the flow rate constraints. Again, let $r \in \mathbb{R}^{m_d \times 1}$ be the vector of flow rates and $I$ the identity matrix. These constraints can be represented as

$$Ir \geq 0.$$

**Proposition 2.** The convex polyhedron defined by

$$\begin{bmatrix} C \\ -I \end{bmatrix} r \leq \begin{bmatrix} b \\ 0 \end{bmatrix}$$

is nondegenerate: there is no point $v$ in the $n_d$-dimensional space that satisfies $n_d + 1$ of the $|\Pi_d(F)|$ inequalities with equality.

**Proof.** If such a $v$ exists, then one of the $n_d + 1$ inequalities that it satisfies is redundant. But $\Pi_d(F)$ was computed by eliminating all redundant constraints, a contradiction. \(\square\)
Because the convex polyhedron is nondegenerate, we can use the pivoting algorithm proposed by Avis et al to enumerate the vertices of this polyhedron (see [2]).

### 3.2.3 Redundancy Testing

The remaining task is to test if all the vertices lie in the same halfspace.

**Definition 1.** We say that an $n$-dimensional vertex $v$ lies in the halfspace defined by $c$ (the halfspace is $\sum_{i=1}^{n} c_ir_i \leq b$) if and only if $\sum_{i=1}^{n} c_iv_i \leq b$.

### 3.2.4 Secure Two-Party Scalar Product Protocol

Goethals, et al. describe a secure two-party scalar product computation protocol in [6] using the properties of homomorphic encryption. The protocol solves the following situation: Alice has a vector $x$ and Bob has a vector $y$. Alice, but not Bob, would like to get the result of $x \cdot y$ without Bob finding out what $x$ is, and Bob does not want Alice to know $y$. We refer to this secure two-party scalar product protocol as S2PSP.

### 3.2.5 Resource Management Algorithm

This leads naturally to the following protocol for computing the minimal resource state abstraction. See Algorithm 1.

### 4 Implementation

The implementation of SFP is not yet complete, so the following sections will outline the ongoing work being done to implement and evaluate SFP. Additionally, this section will describe the implementation and evaluation of Unicorn, a unified resource orchestration system for multi-domain, geo-distributed data analytics [11]. While SFP does not incorporate job orchestration, Unicorn uses the same resource state abstraction framework described above when scheduling jobs and thus provides a way to evaluate the RSA framework described. Figure 5 shows the Unicorn system; the “ReSA” (resource state abstraction) components compute the minimal representation of their resource limitations in a distributed way, and share the results with the global resource orchestrator.
Algorithm 1 A secure, decentralized protocol for minimal resource state abstraction.

1: Each domain $d$ computes its resource state abstraction $\Pi_d(F)$.
2: Enumerate the vertices of the convex hull defined by $\Pi_d(F)$ using Avis’s pivoting algorithm.
3: if a flow $f_j$ does not enter $d$ then
4: Assign a large number to the $j$th dimension for all computed vertices.
5: end if
6: for $c \in \Pi_d(F)$ do
7: for domain $d' \in D$ do
8: $d$ uses S2PSP to test if all vertices in $d'$ are in halfspace defined by $c$.
9: if all the vertices in $d'$ are the halfspace defined by $c$ then
10: $c$ is redundant to the constraints in $d'$.
11: else if all vertices in $d'$ are in the complementary halfspace of $c$ then
12: $\Pi_d(F) = \emptyset$ and $\Pi_D(F) = \emptyset$.
13: Domain $d$ floods this message to other domains and the application.
14: else
15: Constraint $c$ is not redundant to $d'$.
16: end if
17: end for
18: if $c$ is not redundant to at least one other domain then
19: Constraint vector $c$ is not redundant.
20: Add it to $\Pi_d(F)$.
21: end if
22: end for

4.1 SFP and ESnet

We are communicating with the Energy Sciences Network, known as ESnet, a high-performance network built by the United States Department of Energy to support science research at the US National Laboratories and other American research institutions. In 2014, around the time of LHC’s redesign, ESnet deployed four high-speed transatlantic links with a total capacity of 340 Gbps and with European equipment connected by 100 Gbps links provided by GÉANT [3]. This links Brookhaven National Laboratory and Fermi National Laboratory, which are Tier 1 sites for CMS and ATLAS, respectively, directly to the LHC in Geneva. This connection extends the work of Harvey Newman of Caltech, who led the development of US LHCNet to connect CERN directly with American laboratories [4]. For these reasons, ESnet presents a good setting for evaluating SFP. It has close connections to the CMS experiment, and involves many domains interacting in a federated way.
4.2 Unicorn and Resource State Abstraction Demonstration

A demonstration of the Unicorn system was presented at SC17, the ACM/IEEE Supercomputing conference, in Denver, Colorado. We traveled to Caltech’s campus in Pasadena, California a few weeks prior to SC17 to create a demo topology to highlight Unicorn’s resource discovery and job orchestration capabilities.

4.2.1 Demo Topology

We created a network topology linking together the Caltech/inqnet booth at SC17, a testbed of DTNs and switches in the Downs-Lauritsen Physics Laboratory at Caltech’s main campus, and additional computation and storage nodes at the UNESP and 2CRSI booths at SC17. See Figure 6.

In the demonstration, we ran OpenDaylight and Kytos, two different SDN controllers, in the different domains, to underscore how the Unicorn protocol enables multi-domain coordination in a way that respects privacy.

4.2.2 Demo Results

Unfortunately, we experienced problems connecting the demonstration booth to the Caltech testbed via SCinet, the network provider of SC17. So, we were unable to
complete the interdomain demonstration of the resource state abstraction. Nevertheless, we were able to simulate interdomain resource state abstraction by running a Mininet topology on the switches in the Caltech testbed located in Pasadena. Using the Mininet topology, Unicorn optimally scheduled the tasks it was asked to, and the participating “domains” successfully shared their resource abstractions in a way that did not compromise network privacy.

5 Next Steps

There are some areas that deserve further research before SFP can be fully deployed and tested.

5.1 Convergence

It can be shown that BGP policy-routing converges because of the economics of the customer-provider relationships among Internet Service Providers. SFP and Unicorn were designed in the context of federations of scientific networks (WLCG, CMS, ESnet, etc.). Although SFP has very broad applications that extend to commercial networks in addition to science networks as SDN becomes more and more widespread, it will be important to prove that its routing converges.
5.2 Routing Table Implementation

This paper did not fully detail how to store routing and forwarding tables that include multiple dimensions of rules in a way that does not lead to a cross-product explosion. Moreover, we need to further refine the data structures used for prefix matching and lookup. It will be important to consider how these tables can be easily sliced and shared with neighboring domains.

5.3 Merging Pipelines

We also must consider how pipelines from various neighbors will be merged together to form a given domain’s routing information base and how possibly overlapping rules will be chosen when forwarding packets.
References


[17] https://cms.cern/collaboration

[18] https://cmsweb.cern.ch/phedex/about.html
