Toward Interconnection Abstractions to Realize Efficient and Agile SDC Networking

Y. Richard Yang[‡] Franck Le[◇] Christopher Leet[‡] Christian Makaya[◇]
Vinod Mishra[+] Miguel Rio[†] Jeremy Tucker[*] X. Tony Wang[‡]
[+]ARL [*]Dstl [◇]IBM/US [†]UCL [‡]Yale University
vinod.k.mishra.civ@mail.mil, jtucker@mail.dstl.gov.uk
{cmakaya,fle}@us.ibm.com, {christopher.leet,xin.wang,yang.r.yang}@yale.edu

***Abstract*—As software defined networking (SDN) can become a key technique for the control of individual coalition networks of a software defined coalition (SDC), the interconnection of such networks can become a major challenge. Traditional approaches such as BGP to interconnect autonomous networks are designed for less flexible networks and hence can have serious efficiency and flexibility issues. In this short paper, we define the requirements of the interconnection of such networks and provide an overview of a novel protocol, called the SDC-Network Federation Protocol (SFP). Instead of being a traditional push protocol such as BGP, SFP adopts a novel pub-sub model to substantially increase efficiency and flexibility. Going beyond only packet handling, SFP introduces flexible network information spaces, such as the packet space and the flowset space.**

I. Introduction

Given the significant potential benefits of software defined networking (SDN) in realizing the vision that a network can be effectively controlled using a simple, logically centralized control-plane program with a global view, it is natural to consider that individual coalition networks of a software-defined coalition (SDC) may evolve to introduce SDN as a key technique. As a result, the interconnection of such SDN networks will become a major requirement, to realize benefits including expanded reachability and pooled resources.

The interconnection of such new networks, however, can be challenging, and existing work (*e.g.*, SDX [1], SD-WAN) focusing on such new networks considers limited settings such as the existence of a third party. Since some might think that one can still use traditional interdomain protocols to interconnect SDC networks, consider the issues when applying BGP, the well-developed, *de facto* interdomain protocol. Unfortunately, applying BGP to such interconnection can have fundamental mismatches. For example, designed in a traditional networking setting with limited programmability, BGP is fundamentally a *full instantiation* information-exchange protocol, in that the program decisions at each network need to be fully instantiated as data and then exchanged among networks. The extremely large decision space of SDN, and in particular, the two-layer SDN decision model, where the routing information base layer is only a cache of the SDN program layer, can make full instantiation unfeasible.

The objective of this short paper is to report our progress on conducting a systematic investigation and design of interconnecting SDN networks of an SDC. Following the approach that *abstractions based on decomposition (modularity) is the way problems are solved* [2], we focus on key abstractions. For more details, please see [3], [4].

II. Requirements

We identify the following requirements:

- *Efficiency*: The abstractions introduced in the design should achieve efficient integration of resources spanning across multiple networks. The efficiency includes both the protocol, which includes the scalability, latency and stability of the protocol messages, and the outcome of the protocol, to achieve global optimality under constraints. The resources should include memory, network, and compute resources.
- *Autonomy*: There is a trade-off between autonomy and stability. In an SDC where each coalition member network can implement its own policy, policies from different members may conflict and result in instabilities. The design should offer a large degree of autonomy but also consider stability.
- *Privacy*: Although one coalition member network may send multiple queries to another member, the first member should not be able to learn information that the second considers sensitive and chooses not to reveal.

III. Architecture

The model-views abstraction framework: Considering SDN as a core concept, we design interconnection abstractions in a unified model-views framework, where the logically centralized control program P_A of network A defines the core model. To be concrete, we show an example P_A:

```
f(Packet pkt):
  if (pkt.tcpSrcPort != 80)
    return drop
  else
    srcClass = map_policy(pkt.Ipv4Src)
    return shortestPath(srcClass, pkt.Ipv4Dst)
```

It is from this model that multiple abstraction views are derived, where the views include the north bound (NB), which is the interface of the network to applications using the

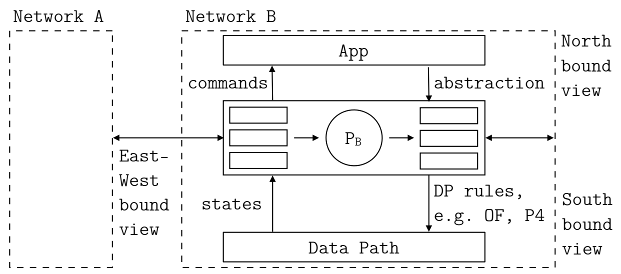

Fig. 1. The unifying model-views abstraction framework.

network; the south bound (SB), which is the realization from the control plane to the data plane; and the east-west bound (EWB), which is the interface between peering networks and hence is the focus of this paper. See Fig. 1 for illustration of this unifying framework.

Adopting a unified model-views abstraction framework leads to multiple benefits, including automation (as views should be derivable from the model), sharing of common functions when computing multiple views, and consistency (among views, as implied from their consistency to the model).

Interconnection information organization and exchange: In a high level, what a network B provides to another network A to achieve interconnection is how P_B will handle each packet that A is interested in. In the basic level, P_B processes each packet independently. Hence, the basic information that B needs is how each packet in a *packet space* is handled. As coalition networks integrate more functions such as QoS and security, P_B classifies packets into flows. Since the processing of flows can be correlated, in particular, in resource sharing case, we say that A may request information for a set of flows. We refer to this as the *flowset space*, where each point in the space is a set of flows. In the general case, we allow extensible information spaces, with two predefined spaces: the aforementioned packet space and the flowset space.

With the basic concepts of information spaces, we specify the basic message flow, as illustrated in Fig. 2. It is a sub/pub protocol, where a network A sends a sequence of subscription interests to network B. Each subscription is for a subspace of an information space that B supports.

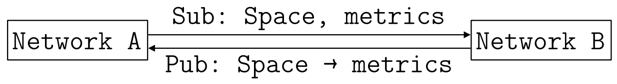

Fig. 2. Basic message flow.

The packet space abstraction: The packet space is the most basic space and hence we give an overview of how we compute abstractions for it. The flowset space abstraction will be presented in a companion short paper.

We define the packet space abstraction problem as the following compact policy program instantiation (CPPI) problem: given (1) a program P, which takes a point in packet space and returns a routing decision for that point's packets, and (2) a queried subset of packet space, Q, what is the most compact instantiation of P's behavior over Q? Such an instantiation will be used as basic information to communicate an network's controller's program to other networks in response to queries.

Although the CPPI problem is challenging, we have conducted a systematic study of the problem while ensuring correctness, updatability, mergability and obfuscation by abstracting P's behavior over subsets of packet space as pipeline tables. Fig. 3 presents the architecture of our approach.

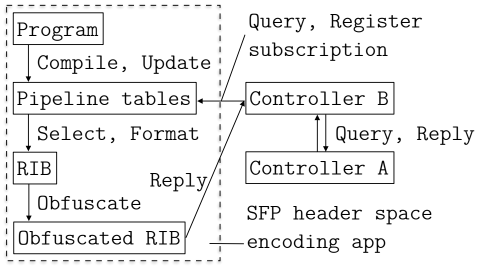

Fig. 3. Packet space abstraction architecture.

The key step of the packet space abstraction is program compilation, which is the transformation (*i.e.*, abstraction) of a generic program, expressed in a high level language, into a pipeline of flow tables. Programs are compiled into flow tables because flow tables allow efficient querying and require little additional processing to be transmitted efficiently. Compilation occurs "only-once", offline, avoiding the burden of per request program compilation and removing a potentially lengthy computation from the runtime environment.

Fig. 4 illustrates the compilation workflow. Specifically, we first compile a program into a per instruction table (PIT) pipeline using two synergistic techniques: symbolic map and flow explore. Subsequently, we compress PIT into a compact representation via a third technique, deep expansion. In even simple settings, we can reduce information exchange size by x1000 compared with single table abstraction (*i.e.*, BGP).

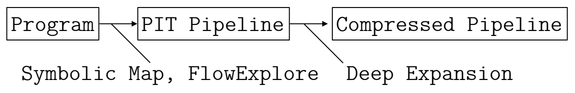

Fig. 4. Program to pipeline abstraction compiler architecture.

Acknowledgement

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] A. Gupta, L. Vanbever, M. Shahbaz, S. P. D. B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "SDX: A Software Defined Internet Exchange," in *Proc. of SIGCOMM*. IEEE, August 2014, pp. 233–239.
[2] B. Liskov, "The power of abstraction," in *Proc. of International Symposium on DIStributed Computing*, September 2010.
[3] F. Le, C. Leet, C. Makaya, M. Rio, X. Wang, and Y. R. Yang, "SFP: Toward a scalable, efficient, stable protocol for federation of software defined networks," in *DAIS Workshop 2017*, August 2017.
[4] A. Voellmy, S. Chen, X. Wang, and Y. R. Yang, "Magellan: Generating high-quality multi-table datapath from datapath oblivious algorithmic SDN policies," in *Proc. of ACM SIGCOMM (poster)*, August 2016.