Journal of Network and Computer Applications 176 (2021) 102947

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

A survey on network forwarding in Software-Defined Networking

Liang Yang *, Bryan Ng *, Winston K.G. Seah, Lindsay Groves, Deepak Singh

School of Engineering and Computer Science, Victoria University of Wellington, New Zealand

ARTICLE INFO

Keywords:
Software-Defined Networking (SDN)
Forwarding table entry (FTE)
Forwarding pipeline
Forwarding representation
Networking element
Boolean algebra

ABSTRACT

The packet forwarding behaviour of a network relies on the forwarding rules residing in the networking element (e.g. switches and routers) to forward packets. This applies to both traditional networking and software-defined networking (SDN). These forwarding rules vary in their structures and sizes, but they represent the same fundamental functionality: to match incoming packets against rules and determine its forwarding behaviour. All these rules in a network device constitute a packet forwarding pipeline. This survey paper presents a new taxonomy for the representation of the forwarding pipeline in SDN based on core networking functionality. The new taxonomy uses a typology and attributes of the forwarding behaviour to establish a cohesive vocabulary and a classification system to study and characterise the interactions of FTE along an end-to-end path in SDN. The survey also shows that Boolean algebra is most widely used for the low-level representation of FTEs and facilitates the assessment of "equivalence" between FTEs and has practical uses in traffic monitoring and packet classification within SDN, among others.

1. Introduction

Networking elements in a computer communication network are forwarding devices (switches or routers) linked together to deliver packets from one point to another. In a traditional network (e.g. Internet Protocol (IP) and MPLS (Multi-protocol label switching), these networking elements have a control plane and a data plane coupled together (Masoudi and Ghaffari, 2016). The control plane performs control functions such as deciding the best routing (or forwarding) path, installing rules for forwarding and configuration of networking elements along the end-to-end path. While the data plane forwards packets based on forwarding rules made by the control plane.

The forwarding rules in traditional networking are defined as access control list (ACL) and IP lookup table. The ACL is mostly configured by the administrator manually while IP tables are usually created and updated locally by the routing protocol residing in each networking element (Habib et al., 2019; Kao et al., 2019). Often, the ACL and IP tables are modified manually by network administrators to manage localised requirements resulting in a complex network configuration. In addition, the use of a low-level programming language and vendor dependency makes configuration of the network cumbersome and error prone for network operators.

To simplify the configuration complexity in the traditional network, a new networking paradigm called Software-Defined Networking (SDN) that uses a high-level programming language has emerged (Nunes et al., 2014; Zhang et al., 2018a). SDN is a network paradigm in which the control plane is decoupled from a networking element. In an SDN, the control function is moved to a logically centralised controller which eases the load on a networking element with the primary function of forwarding packets. It allows the network to be controlled and managed through a software written in a high-level programming language.

The shift of the control function from a hardware to software dependency allows the system to be more agile, programmable and centrally managed. The forwarding rules in an SDN are defined by a set of flow table and group tables, the difference between flow tables and group tables is that Group tables consist of multiple flow table entries and actions associated with particular groups (as defined by the controller). In the remainder of this paper, we generalise flow tables and group tables as forwarding table entries (FTEs), except where it is necessary to differentiate them.

Forwarding table entries have more flexible structure, stronger forwarding manipulation abilities and inevitably more complicated logic than the ACL and IP tables. These dynamically provisioned FTEs control packet forwarding behaviour from a network-wide perspective rather than the device-level perspective in ACLs and IP tables (Levin et al., 2012). To model the network-wide forwarding behaviour, a controller must have a systematic method to represent FTEs across the network. Moreover, a common vocabulary and representation (or model) of FTEs is necessary to systematically model end-to-end network behaviour and

* Corresponding authors.
E-mail addresses: yzyangliang@gmail.com (L. Yang), bryan.ng@ecs.vuw.ac.nz (B. Ng), winston.seah@ecs.vuw.ac.nz (W.K.G. Seah), lindsay@ecs.vuw.ac.nz (L. Groves), deepak.singh@ecs.vuw.ac.nz (D. Singh).

https://doi.org/10.1016/j.jnca.2020.102947
Received 25 August 2019; Received in revised form 5 October 2020; Accepted 7 December 2020
Available online 9 December 2020

abstract low-level-behaviour. In this survey we show that FTEs serve as an appropriate abstraction for end-to-end network behaviour.

The motivation to investigate the existing technologies on forwarding pipelines which refers to the functionalities of networking elements, forwarding rules and the topologies connect these elements. These functionalities determine the forwarding path of packets which are similar to the pipelines in a watering system, hence also called as ***packet forwarding pipelines***.

In this survey, we will review the literature on network forwarding and representation of forwarding pipelines. This survey offers an overview on the state-of-the-art advances in representations of the packet forwarding pipelines that contributes to the understanding of key features of a packet forwarding behaviour. This survey reviews existing and up-to-date technical solutions, identifies their basic characteristics to derive the essential FTE attributes which must be covered by an FTE representation.

The goal of a representation is to better understand and utilise the functionality of FTEs inside SDN. An effective FTE representation must meet the following criteria: (i) it should cover the major attributes of FTE, for example, wildcard, priority and multi-table; (ii) it should be capable of exploiting the various manipulations on FTEs such as validation and facilitating FTE placement; and (iii) it should have the ability to integrate network information such as traffic statistics and topology to resolve realistic networking problems. With the help of the representation, a controller can easily transfer high-level policies to low-level FTEs and correctly place them into switches or routers.

The scope of this survey includes FTE representation and networking applications based on the analysis of different approaches to characterising FTEs. In the review of a FTE representation, the forwarding pipeline in traditional networking and SDN are inspected and characterised which forms the basis of the taxonomy we develop later in the survey. Moreover, the various representations to analyse FTE (called a typology — grouping of FTE representations with shared similarities) as well as the rationale why they are chosen and their respective applicable scenarios are presented. To better relate the outputs of this survey to practice, we review two types of applications: (i) assessing equivalence in forwarding behaviour and (ii) traffic monitoring.

An overview of SDN is given in Section 2 to guide readers through the concepts, architecture, and the de-facto protocol (OpenFlow) of SDN. It is then followed by the deconstructing the forwarding pipeline in both the traditional network and SDN in Section 3. This is followed by the core contribution of this survey which is a taxonomy of FTE representation in Section 4. Next, the taxonomy is applied to systematically categorise the literature with comments related to why they are suited and their respective application scenarios in Section 5. Finally, a summary of this survey is given in Section 6.

2. SDN: Overview

In a networked system, network elements have two basic components: the control plane and the data plane. The control plane performs control logic such as routing protocols, middlebox configuration while the data plane forwards the traffic based on control logic. The control plane and data plane are coupled together, thus enabling each network element to participate in route making decisions and data forwarding.

However, this tight coupling is not flexible when the network size increases. The obfuscated control logic coupled with proprietary hardware and the use of low-level programming languages makes the system rigid and complex. Therefore it copes poorly with issues of scalability, reliability and security which are more pronounced as the network size increases. These issues in the traditional network hinder the performance of the system as the network traffic is increasing by the day (Saraswat et al., 2019; Lu et al., 2019; Hossein et al., 2019; Kuan and Dimyati, 2006; Hsueh et al., 2018).

To cope with the aforementioned issues in the traditional network, a network paradigm called "Software-Defined Networking" (SDN) was seen as the next generation of computer networking. SDN has two defining characteristics that sets it apart from traditional networks: (a) it separates the control plane from the data plane, and (b) it logically centralises the control plane to direct multiple data plane elements through a single software control program (Feamster et al., 2013). From SDN, new areas of research such as Network Function Virtualisation (NFV) have emerged, network service chaining, network as code have emerged (Yan et al., 2018; Li et al., 2019; Zhou and Benson, 2019; Sanger et al., 2019; Qiu et al., 2019; Lin et al., 2019; Vardi and Kupferman, 2019; Kyung and Park, 2019; Riener et al., 2019).

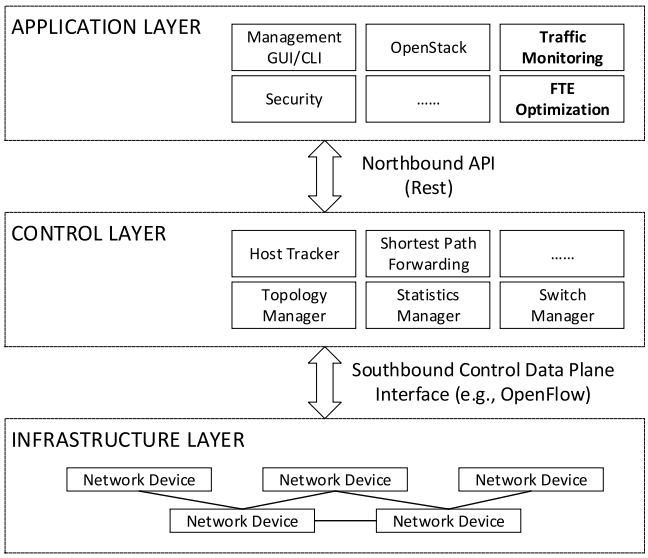


Fig. 1. Simplified SDN Architecture.

A simplified architecture view of an SDN is depicted in Fig. 1. The core of an SDN enabled network is the controller which not only exercises direct control over all networking devices, but also responds to the requests from the application side. A controller communicates with the higher-level components, the applications, via the northbound interface. Similarly, it also communicates with the lower-level components for example the network elements across the network, via the southbound interface.

Some protocols for the southbound interface are OpenFlow (McKeown et al., 2008), ForCES (Haleplidis et al., 2015), Border Gateway Protocol (BGP) (Rekhter et al., 1994), Network Configuration Protocol (NETCONF) (Enns et al., 2011), Open vSwitch Database Management Protocol (OVSDB) (Pfaff et al., 2015; Hao and Ng, 2019), OpFlex (Smith et al., 2014). OpenFlow is among the first and most widely used protocol to define communication between the controller and switch in an SDN paradigm (Goransson and Black, 2014; Hao et al., 2017; Li et al., 2016; Hao and Ng, 2018).

Traditional network management protocols such as NETCONF or routing protocol such as BGP have been reused to configure the switches in SDN. Even though these protocols do not provide the same flexibility as OpenFlow, they are considered as southbound APIs and supported by major vendors such as NEC, Pica8 and HP. Some alternatives to OpenFlow have been documented in the literature for example OVSDB, ForCES and OpFlex. OVSDB manages Open vSwitch implementation by adopting a non-OpenFlow protocol to program OpenFlow switches. Similar to OpenFlow, both OpFlex and ForCES follow the policy-driven mechanism. OpFlex replaces the OpenFlow controller and OpenFlow interface with "Policy Authority" and "Policy Agent", respectively. However, the switch in OpFlex still relies on the FTE-like flow tables to forward packets.

As a competing protocol with OpenFlow, ForCES is different from OpenFlow in many aspects, but they share the same design principle in

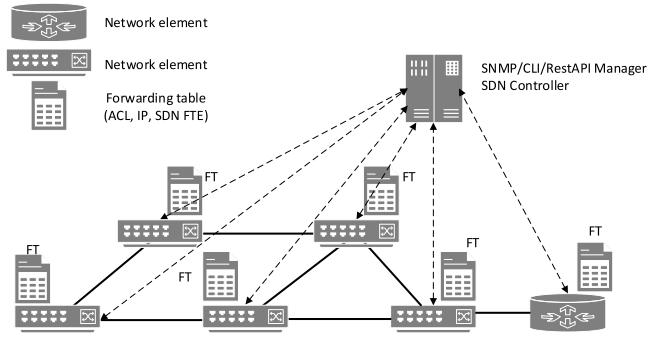


Fig. 2. A network abstraction: forwarding tables.

forwarding models. ForCES manages the packet forwarding behaviour with an abstraction of Logical Functional Blocks (LFBs) which share the similar functionality as OpenFlow FTE. Both LFBs and OpenFlow tables consist of "Match-Action"-like entries, i.e. FTE. Therefore, the underlying principles of an SDN forwarding discussed in this survey extends to non-OpenFlow protocols too.

Recent surveys have shown that the proliferation of SDN across different domains is widespread (Li et al., 2018; Yu et al., 2018; Rojas et al., 2018; Alsaeedi et al., 2019; Omar and Samsudin, 2019; Souri et al., 2019; Su et al., 2019; Ben Azzouz and Jamai, 2019; Kantor et al., 2019; Xie et al., 2018; Sultana et al., 2019; Liu et al., 2019) even reaching emerging areas such as machine learning (Sultana et al., 2019; Xie et al., 2018; Bakker et al., 2018), Internet of Things (IoT) (Salman et al., 2018; Priya and Silas, 2019; Farris et al., 2018) and blockchains (Yang et al., 2019; Jindal et al., 2019). Our extensive research of the literature has identified that none of the surveys in the recent two years have collated and studied FTE representations and classified them as we have done in this survey.

3. Forwarding pipeline

In networking, a pipeline is a chain of packet-processing entities (or filters) connected in a certain type of structure, where the output of one entity is the input of another one. For an individual networking device (switch or router), the packet-processing entities include the various chained forwarding tables such as the ACL table, IP routing table in traditional networking and generic forwarding table in SDN. Similarly, for a network (traditional or SDN), the packet-processing entities include forwarding tables in all networking devices which are connected with a certain type of topology. This pipeline is also called ***packet forwarding pipeline*** because its major function is forwarding data packets between two devices.

3.1. Forwarding tables in traditional networking and SDN

As illustrated in Fig. 2, a packet's forwarding path is determined by the forwarding tables in each network element. The forwarding tables in traditional networking are usually statically configured or generated by locally running routing protocols. Whereas, the forwarding tables in SDN can be dynamically provisioned and updated by a centralised controller.

Traditional networking mainly relies on routing and switching protocols such as BGP (Border Gateway Protocol), IGRP (Interior Gateway Protocol) and Ethernet to perform packet forwarding. However, some applications such as firewall and load balancing need more flexible ways to dynamically regulate the network. To meet this requirement, the concept of an ACL was introduced by Cisco to provide an alternative to manipulate the switch forwarding behaviour. An ACL can be considered as a special purpose filter in a forwarding pipeline which is designed to specify the access rights allowed or denied for all incoming packets.

Besides an ACL table, an IP routing table has also been widely used in a forwarding pipeline (Huang and Zhao, 1999). In a network, each networking device maintains a routing table which consists of a set of IP prefix entries and their associated egress interface(s). When an IP data packet reaches a device, this packet's destination address will be matched against the routing table to find its egress interface(s) by matching against the longest IP prefix. Compared to an ACL, an IP routing table is a generic purpose filter and the most important components of a forwarding pipeline. IP routing tables determine a packet's forwarding path in a network.

From the above two functions, we can see that ACL policy and IP routing in tandem with the underlying protocols perform the following two steps to achieve "network management": (i) the incoming packets match against ACLs or IP route (e.g. prefix matching) and (ii) execute certain actions (drop or output packets). This specific design for each ACL and IP routing function is radically changed by use of forwarding tables in SDN (Benson et al., 2009).

The core concept of an SDN is that by creating a few careful abstractions based on ACL and IP routing tables, the complexity of the underlying components is hidden and the new functionalities are easily developed via the programming interface between the network devices and the controller. One popular programming interface is OpenFlow which is the de-facto southbound interface that gives the higher level software access to the forwarding plane of an SDN. In OpenFlow, FTE has replaced the ACL and routing entry of traditional networking to manage the forwarding capabilities of networking devices. Packets are forwarded according to the path which is composed of FTEs. For a pure SDN, the behaviours of all packets can be precisely controlled by manipulating FTEs.

OpenFlow was proposed around 2008 and it is still evolving as the mechanism and usage of FTE are not fully researched (Pfaff et al., 2009). The core idea of FTE is similar to the policies which have been widely used in an ACL while the policy's scope and depth have been largely extended in OpenFlow. As the network scales and grows in complexity, FTEs are not easy to understand and manipulate. Therefore, it is important to find a way to represent FTEs as they determine the packet forwarding behaviour in an SDN.

The two major perspectives of OpenFlow specification are FTE manipulations (installation, deletion, modification) and statistics updates (Dargahi et al., 2017). These perspectives motivate the research on FTE representation and manipulation for better management of an SDN. In the following subsection, existing research works on characterising forwarding behaviour are discussed. The significance of FTEs and how they relate to optimal network utilisation has been discussed in the works of Guo et al. (2018, 2017) whereby the relationship between flow table occupancy and link utilisation is established.

3.2. Existing works on characterising forwarding behaviour

Existing works that investigate the forwarding behaviour of packets can be classified into three categories based on three different forwarding tables whose structures and functionalities vary. The three forwarding tables are (i) ACL, (ii) IP routing entry, and (iii) FTE. This subsection reviews the existing works for all these three forwarding tables and derives the four essential attributes for a successful forwarding pipeline representation.

3.2.1. Regulating forwarding behaviour by ACL

An ACL is composed of a sequence of rules to match against the packet to determine whether a specific action is performed or not. It is a special type of role-based access control (RBAC) (Ferraiolo et al., 2003). A "minimal RBAC Model" (RBACm) can be compared with an ACL mechanism (ACLg) where only groups are permitted as entries in the ACL (Barkley, 1997). An ACL has been widely used in a computer

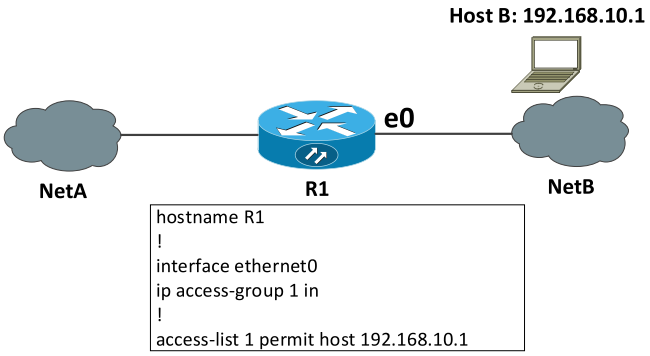


Fig. 3. An ACL example with corresponding access matrix.

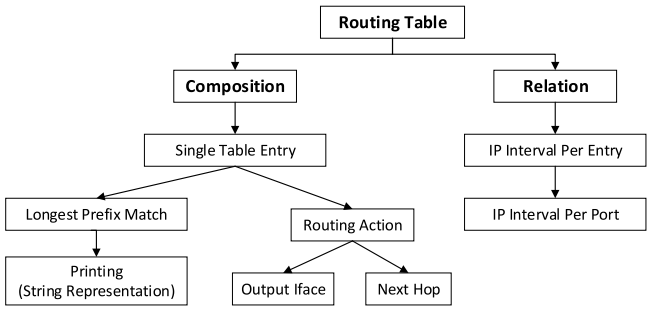


Fig. 4. IP routing table structure.

Table 1
Access matrix for the ACL example in Fig. 3. The access matrix resides in router R1.

Subject	Object	Operation
Interface ethernet0 (e0)	Incoming packets	Permit host 192.168.10.1
*	*	Deny

Table 2
"Match-action" form for the ACL example in Fig. 3.

Match Fields	Action
Ingress port = Ethernet 0 Destination IP address = 192.168.10.1	PERMIT
*	DENY

system such as the user account authentication, firewall, etc. The prior research on ACL is mainly focused on the validation of these access control policies and the optimisation of their storage space.

Lampson introduced the formal notions of *subject* and *object* as well as an access matrix that mediated the access of subjects to objects (Lampson, 1974). An *access matrix* consists of a set of subjects $s \in S$, a set of objects $o \in O$, a set of operations $op \in OP$, and a function $ops(s, o) \subseteq OP$, which determines the operations that subject s can perform on object o.

To model a simple network as an access matrix, we use an example of a network enforcing an ACL. Fig. 3 shows an ACL example where a select host in NetB is granted permission to access NetA. All traffic from Host B destined to NetA is permitted while all other from NetB destined to NetA is denied. Its corresponding access matrix is demonstrated in Table 1. The specified operations (permit host 192.168.10.1) are performed on all incoming packets which are the implicit objects of an access matrix.

An access matrix can be easily transformed to the "match-action" form (MAF) which has been specified in OpenFlow specification (ONF, 2014). For an example in Fig. 3, its equivalent MAF is illustrated in Table 2. An MAF does not include the field of object in access matrix because all ACLs share the same object: incoming packets. The action in MAF only contains two operations: permit and deny.

The default action for all packets which are not explicitly permitted by ACLs is denied which is why the second row is added in Table 2. The match field in a MAF is composed of an access matrix's subject and the negatives of the fields in operation. From the perspective of functionality, an ACL is equal to a generic forwarding table with no constraints on match fields and only two type of actions (permit and deny) allowed.

3.2.2. Regulating forwarding behaviour by IP routing table

An IP routing table determines which specific route is selected for a given IP address. The table entry in an IP routing table represents the smallest subnet that contains the given IP address. A routing table captures two aspects: Composition and Relation (Michaelis and Diekmann, 2016). The Composition aspect deals with the components of a routing table and their respective meaning while Relation interprets a routing table from different functional perspective. The structure of an IP routing table with the Composition and Relation aspects is illustrated in Fig. 4.

As seen in Fig. 4, a routing table is composed of single table entries where each entry contains a longest prefix IP address and its corresponding routing action (a subset relationship). The relation between an IP address and its corresponding routing action can be then interpreted from the two functional perspectives: (1) IP interval per entry and (2) IP interval per port. The IP interval per entry is a function from IP (interval) to port (where the intervals do not overlap) while the IP interval per port is a function from port to IP (a range of IP addresses or an IP space). Algorithm 1 defines the routing table semantics (Michaelis and Diekmann, 2016) in which the datatype *linord-helper* maps the set relationship between two IP addresses (lines 2–6). It is used to express the concept of longest prefix match (LPM) (lines 13–16). For all IP packets which match the given routing table entries (lines 8–11), they will be forwarded to the corresponding output interface or next-hop depending on their associated routing-actions (lines 17–19).

3.2.3. Regulating forwarding behaviour by FTE

OpenFlow provides an industry-standard application programming interface and protocol to program forwarding tables in switches. OpenFlow is managed by Open Networking Foundation (ONF), an organisation dedicated to promoting and adoption of SDN. OpenFlow is evolving and will continue to evolve. The latest OpenFlow switch specification is Version 1.5.0 (Protocol version 0 × 06) (ONF, 2014) which was released on December 19, 2014 by ONF.

Since OpenFlow Version 1.1.0 (ONF, 2011), a flexible pipeline with multiple tables is exposed to the control layer. As depicted in Fig. 5, packets are processed through a pipeline which consists of one or more flow tables, group tables whereby each table consists of multiple FTEs. The pipeline has two processing stages: ingress processing and egress processing which is separated by the first egress table as seen in Fig. 5 (ONF, 2014). The pipeline processing starts with ingress processing from the first flow table, labelled "Table 0" in Fig. 5. The output of ingress processing may direct a packet to an output port in egress processing which is optional. The incoming packet "Packet In" must be matched against FTEs of "Table 0" and subsequently the packet might be forwarded to any table following "Table 0" based on the action set and pipeline instructions such as metadata which carries information between tables. When a packet is matched against the FTEs of a flow table, a matched flow entry is selected to execute a set of instructions.

The instructions can be to forward the packet to a specific port or another table. The instructions can also direct the packet to a Group table for further processing as seen in Fig. 5. The group table is used to execute group actions on packets and consists of group entries. A Group table contains forwarding entries and actions for packets associated

Algorithm 1 Routing table semantics

```
datatype ('a,'b) linord-helper = LinordHelper 'a 'b
begin
    definition linord-helper-less-eq1 a b ≡ [ case a of LinordHelper a1 a2 ⇒ case b of LinordHelper b1 b2 ⇒ (a1 < b1) ∨ ((a1 = b1) ∧ (a2 ≤ b2)) ]
    definition a ≤ b ⟷ linord-helper-less-eq1 a b
    definition (a ≠ b ∧ linord-helper-less-eq1 a b)
end
theory Routing-Table
import ../IP-Address/Prefix-Match
       ../IP-Addresses/IPv4 ../IP-Addresses/IPv6
       Linorder-Helper
       ../IP-Address/IP-Address-toString
begin
   record(overloaded) 'i routing-rule =
                      routing-match :: ('i::len) prefix-match
                      metric :: nat
                      routing-action :: 'i routing-action
   record(overloaded) 'i routing-action =
                      output-iface :: string
                      next-hop :: 'i word optio
end
```

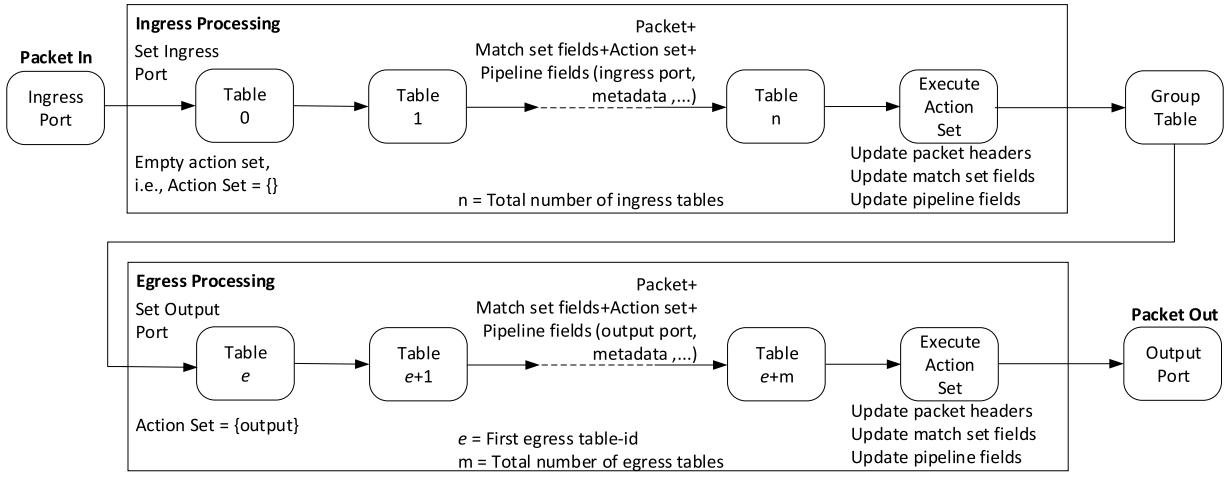


Fig. 5. OpenFlow table processing (ONF, 2014).

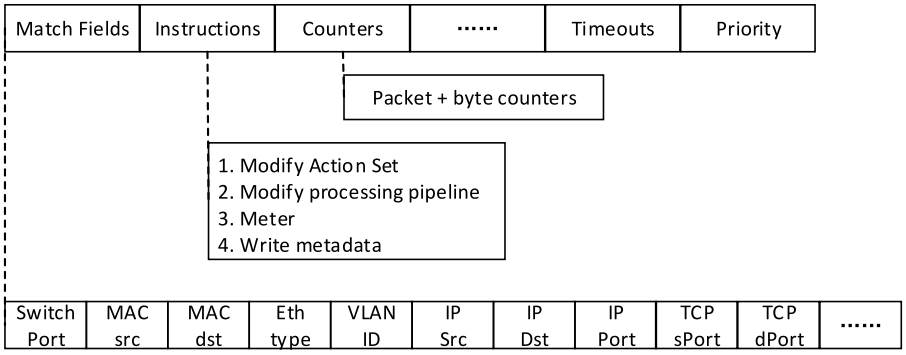


Fig. 6. Inside FTE: OpenFlow packet matching and actions.

with a group, which can be a set of TCP ports, or a set of actions applied to traffic aggregates. The Group table functionality represents an area that is under studied within the SDN context but can be extremely powerful and flexible. Finally, if a packet has no matching flow entry in a flow table, the packet is processed depending on the instructions specified in the table-miss entry.

The match fields and actions in FTE can be easily customised to implement the same forwarding behaviour as ACL and IP routing entry (Bosshart et al., 2013a). For an ACL, the match fields remain the same as FTE, only the ACL's default action "deny" is replaced by "drop". The conversion from an IP routing entry to an FTE is more straightforward, neither the match fields or actions need any changes. But not all match fields in traditional networking are explicitly indicated. For example, the packets with specified VLAN id and destination MAC (DMAC) address will be only sent to an IP routing table for prefix searching. Therefore to maintain the same forwarding behaviour as an existing IP routing entry, the match fields of FTE will become the combination of "VLAN, DMAC".

Table 3
Forwarding table comparison.

Network	Table	Dimensions		Actions			
		Single	Multiple	Copy	Pop/Push	Set	Out
Traditional network	ACL		x			x	x
	IP routing	x					x
SDN	FTE	x	x	x	x	x	x

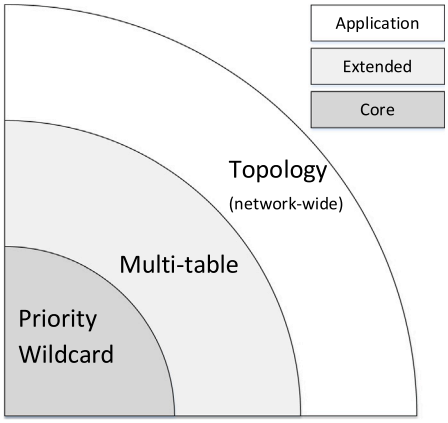


Fig. 7. OpenFlow attributes.

A flow table consists of multiple FTEs. Fig. 6 illustrates the main components of an FTE: match fields, instructions, counters, timeouts and priority. An FTE is identified by match fields and priority. The match fields determine whether a packet can hit an FTE while the priority determines whether a packet has the chance to match against this FTE or not. Only the matching FTE with the highest priority will be selected. A match field contains the well-known fields in an IP header (source MAC, destination MAC, source IP, destination IP, etc.) as well as the fields related to pipeline processing, for example, ingress port and metadata.

From the research works (Yang et al., 2016a, 2017a,b) on SDN forwarding table, there are four essential FTE attributes: (i) Wildcard [W], (ii) Priority [P], (iii) Multi-table [M], iv) Topology [T]. The attributes can be extended by taking into consideration whether an FTE supports group actions or not, however this is beyond the scope of this survey because it is a feature that has not been widely studied at the time this research was conducted.

Among the above key attributes, wildcard and priority are only associated with a single table. Wildcard and priority are inherited from an ACL and a lot of research has been done on these attributes (Pozo et al., 2008; Wong et al., 2010). As illustrated in Fig. 7, they are categorised as core attributes and must be covered by all representations. Multi-table is one of the most significant feature in OpenFlow to enrich dynamic configuration and placement of a FTE. Multi-table is listed as extended attributes in Fig. 7 and it must be represented by any network-wide representation. Although topology is not an intrinsic attribute of a FTE, it is still listed here as a crucial attribute because it plays an important role in determining a network-wide packet forwarding behaviour.

3.3. Pipelines: a common denominator for packet forwarding

Inside a single forwarding element (switch or router), there might have multiple forwarding tables where each table serves a different purpose to match their respective specified field(s) against the incoming packets. These tables construct a packet forwarding pipeline which is the common denominator for ACL and IP routing table in traditional networking as well as the generic forwarding tables in SDN. In the next few paragraphs we will compare by way of example the tables

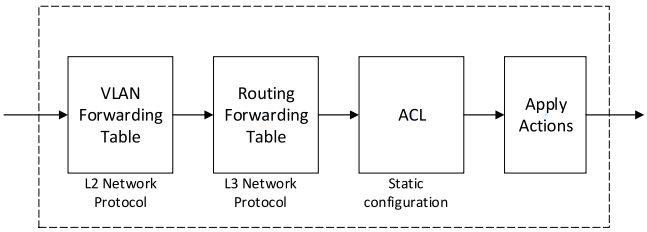


Fig. 8. Switch-level pipeline in a traditional network.

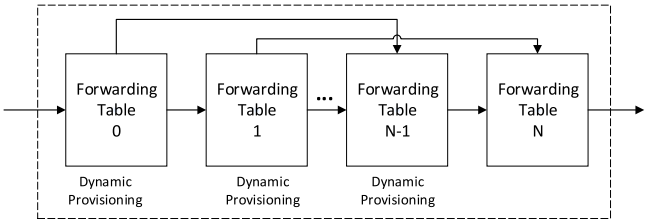


Fig. 9. Switch-level pipeline in SDN.

in traditional networking and SDN to highlight the difference in the forwarding pipeline.

The first example is an IP routing table. It has only one match field i.e., the destination IP address to match upon the incoming packets, it is also called a single dimensional forwarding pipeline (Kang et al., 2013a). The second example is an ACL, which is a multiple dimensional pipeline because it matches against more than one known field simultaneously. However, as the name ACL suggests, it can only decide whether a packet is allowed to pass through a device by its predetermined action: permit or deny.

The SDN paradigm uses a more generic forwarding pipeline which allows arbitrary combination of multiple fields and performs more complex actions. Each of the tables are dynamically provisioned based on the program running in the controller.

In Table 3 we compare the traditional network with SDN. Each table (denoted in the row) is marked with an "x" if the table supports a "Dimension" for matching and is able to perform a set of "Actions". The Dimension column indicates that the table for the use case is single dimensional (e.g. IP or VLAN or TCP) or multidimensional (e.g. TCP and MPLS). For example, IP routing only requires the source and destination address fields from the IP headers (i.e. one dimension) while ACLs must have the capability for regulating access via both IP headers, TCP headers and potentially other headers (i.e. multiple dimensions).

For ACLs, there are many possible combinations of IP addresses or ports the administrator may want to configure. Hence, multiple dimensions are a must for ACL. The Actions are described below:

- copy: copy TTL (Time-to-live) to/from the packet
- pop/push: apply all tag pop/push actions to the packet (e.g. MPLS or VLAN tags)
- set: apply all set-field actions to the packet
- out: forward the packet on the port specified by the output action

Table 3 highlights a stark difference between different table types in traditional network vs. SDN. From the perspective of matching "Dimensions", the IP routing table only has one dimension which is the IP prefix while the rest two types have arbitrary combinations of the fields in a packet header. From the perspective of "actions", the IP routing table always directs to the next-hop via an "Out" action and associates with a single egress interface. For SDN, the tables can implement an arbitrary combination of actions via the FTE, hence all actions are marked "x".

In traditional networking, each table has fixed width and is assigned the corresponding predetermined functionality, for example, VLAN validation in the first table, IP address prefix matching in the second table and ACL filtering in the third table. However, in SDN, these tables can be customised and are not restricted to the predetermined actions. Hence, the ACL and IP routing tables can also be considered as a special type of forwarding pipelines. A forwarding pipeline is composed of one or multiple forwarding tables which contain FTEs to specify the match fields and their associated actions.

Though the structure and functionalities of forwarding pipelines have changed significantly from traditional networking to SDN, the underlying hardware remains the same. Most SDN switches on the market share the same hardware as before. Figs. 8 and 9 illustrate the reference forwarding pipeline of a typical ASIC (application-specific integrated circuit) based switch in traditional networking and SDN, respectively.

4. FTE representation

Even though the OpenFlow protocol is evolving, various approaches have been proposed to analyse and manipulate the FTE in OpenFlow. A summary of the literature of FTE representation is presented in Table 4. FTE representations have been studied using simple logical connectives (Bifulco and Schneider, 2013; Katta et al., 2014; Ng et al., 2013; Hao and Ng, 2019) to more complex and abstract representations such as infinitary logic.

In general, FTE representations are built upon one or a combination of logic. The term "logic" in this survey is an expression system with well defined syntax, semantics and pragmatics. Examples of single-logic based representations include algebra (Foster et al., 2015; Shin et al., 2013, 2012), set theory (Foster et al., 2011; Monsanto et al., 2012), first-order logic based model checking (Son et al., 2013), temporal logic (Al-Shaer and Al-Haj, 2010; Gutz et al., 2012; Wang et al., 2013), and higher-order type theory (Guha et al., 2013a,b). Single logic for FTE representations cannot completely capture the complete attributes for OpenFlow such as the Topology and dynamic table placements. However, single-logic representations are much more tractable and have been the most popular in the literature.

Some representations that adopt more than one logic which includes Z (Shin et al., 2013, 2012) and multiple-logics based model checking (Canini et al., 2012). The Z representation is developed from typed first-order predicate logic and Zermelo–Fraenkel set theory. Multi-logic representations of FTE in SDN are not as well developed compared to single-logic, examples of research applying multi-logic are Skalka et al. (2019), Gordon (2018) and Mycroft et al. (2016).

Our proposed FTE representation is divided into four main typologies (Table 4) namely: logical connectives (LC), algebra (AL), set theory (ST) and formal representations (FR) and each representation has been shown to be able to model one or more OpenFlow attributes described earlier in Section 3.2.3. In this survey, a typology is defined as a grouping of FTE representations with shared similarities. Collectively, the typology and the ability to model out the OpenFlow attributes define the taxonomy of FTE representations in the literature.

4.1. Logical connective (LC)

This typology is denoted as LC in Table 4. ***Logical connective*** representation mainly utilises logical operators to express the relation and interactions between FTEs. The relations that can be in place among match fields and instruction sets are first analysed in Bifulco and Schneider (2013). Based on potential relation combinations, the five FTE interaction types: *Disjoint*, *Exact match*, *Subset*, *Superset*, and *Correlated* are defined in Bifulco and Schneider (2013).

4.2. Algebra (AL)

Algebraic typology is denoted as AL in Table 4. ***Algebraic*** representation is good at describing the attributes and reasoning of a structure or program. The simplest algebra is ***Boolean algebra*** (Whitesitt, 1995) in which the values of the variables are the truth values, *true(1)* and *false(0)*. Another branch of well-studied algebra representation pioneered by E.F. Codd called relational algebra (Codd, 1972), was proposed to model the data stored in relational databases.

Kleene algebra (Kozen, 1997), partly built on relational algebra focuses on the semantics of a program which can be expressed as an idempotent semiring. NetKAT (Anderson et al., 2014; Foster et al., 2015) is another network programming language which is build on top of Kleene algebra has demonstrated its capability of representation for the attributes such as priority and topology. However, there is no evidence to show that NetKAT can also be used to represent multi-table in SDN. A recent paper by Skalka et al. (2019) relates to the concept of algebras to networking in practice and is significant in demonstrating practical application of algebra in SDN.

4.3. Set theory (ST)

Set theory typology is denoted as ST in Table 4. ***Set theory*** (Frenetic Foster et al., 2011) is concerned with the concept of sets. It studies the well-determined collections of objects. However, its strength lies in descriptiveness rather than manipulation, thus it is unlikely to fulfil the desired functionalities related to FTE manipulation, for example, removing the priority in a single table and combining multiple tables.

4.4. Formal representations (FR)

Formal representations is a large family of representations with a long history that predates SDN. Formal representations use rigorous and provable mathematics in the design and implementation of the FTEs. It is a process to derive logical conclusions from the premises which are true or assumed to be true. A formal representation for forwarding pipelines opens up new opportunities for applying formal methods to the analysis, design and optimisation of SDN. Formal representations can be categorised based on the underlying logic used and for representing FTEs these are: (i) symbolic logic and (ii) mathematical logic.

4.4.1. Symbolic logic

Symbolic logic uses symbols and variables to express logical ideas. It is by far the simplest kind of logic. Variants of symbolic logic includes ***propositional logic*** (Clarke, 2012), ***predicate logic*** (Ullman, 1994) and ***temporal logic*** (Emerson, 1990). Among these three logics, propositional logic has no quantifiers while the other two have. The quantifiers are quite useful to express the uncertainties in a forwarding pipeline, such as load balancing and failover (Canini et al., 2012) in which the same packet might be forwarded to different paths.

Propositional logic studies the indivisible statements. It assumes every statement can be interpreted as true or false and then produces more complex statements in which truth-value depends on the truth-values of the simpler statements. The symbols or words used to connect two statements are logical connectives which include binary connectives such as *"or"*, *"and"* as well as unary connectives such as *"negation"*. It has been proven useful in modelling network forwarding pipelines where each entry is interpreted as a logical expression of the conditions to trigger their associated actions (Clarke, 2012).

Since propositional logic is not able to represent the relationship between propositions, a more powerful logic "predicate logic" has been studied in Ullman (1994). Predicate logic is an extension of propositional logic and more expressive. It uses quantified variables such as existential "∃" ("there exists") and universal "∀" ("for all") over

Table 4
Summary of FTE manipulation.

Topology	Logic	Modelled attributes[a]	Applications	Project and Reference
FR	Binary decision diagrams, Computational Tree Logic	[WP]	Identify misconfiguration	RuleChecker (Zhang et al., 2018b), FlowChecker (Al-Shaer and Al-Haj, 2010)
	Rule partition	[WPT]	Scalability	SGLS (Riener et al., 2019),DIFANE (Yu et al., 2011),(Kyung and Park, 2019)
	Propositional logic, temporal logic	[WPT]	Test and verification of OpenFlow applications	FlowLogic (Vardi and Kupferman, 2019), GVCM (Lin et al., 2019), NICE (Canini et al., 2012)
	Computation Tree Logic	[T]	Slicing and Isolation	Slicing abstraction (Gutz et al., 2012)
	Linear temporal logic	[P]	Action-based synthesis of SDN	Automated Synthesis of Controller (Wang et al., 2013)
	Coalgebraic theory, Kleene algebra, Brzozowski derivative	[PT]	All-pair connectivity, Loop-freedom, Translation validation	NetKAT (Anderson et al., 2014; Foster et al., 2015)
ST	Declarative language, Domain specific languages, Set-theoretic operations	[WPT]	A compiler, transformation from application-level policies to switch-level policies transformer	FastRule (Qiu et al., 2019), NetCore (Monsanto et al., 2012)
	Rectangular representation and selection	[WPT]	Rule placement (space, time, resource)	Mozart (Zhou and Benson, 2019), "One Big Switch" (Kang et al., 2013b)
	Brick based model	[WP]	OpenFlow Failure-planning	Semantic (Hsueh et al., 2018), RuleBricks (Williams and Jamjoom, 2013)
	Tree decision model, partition	[WP]	Flow setup efficiency improvement	CAB (Yan et al., 2014a)
	Integer linear programming mode	[T]	Maximise traffic satisfaction	Routing for Efficiency (Nguyen et al., 2014)
	Graph-based model	[PT]	Energy-aware Routing	EAR (Giroire et al., 2014)
	Graph theory, Recursive theory	[WP]	Caching system for SDN, FTE abstraction	CacheFlow (Katta et al., 2014)
	Tree-based model, Tagging approach	[PM]	FTE compression	Generalised FTE Optimisation (Marsh, 2015)
LC	Set theory	[WPT]	Application oriented network programming language over controller	PCNC (Skalka et al., 2019), Frenetic (Foster et al., 2011)
	Logic connective	[WP]	Generic analysis, management and optimisation of FTE	OpenFlow rules interaction (Yang et al., 2017a; Zeng et al., 2014; Bifulco and Schneider, 2013)
AL	Algebra of communicating shared resources	[P]	Formalisation and verification of SDN framework	Formal Specifications (Shin et al., 2012, 2013; Bakker et al., 2019, 2018)
	Tree based model, partition	[WPT]	Traffic-optimal rule placement	BigMAC (Yan et al., 2018), vCRIB (Moshref et al., 2013)
FR, AL	Tree composition/decomposition	[M]	Multi-table processing on legacy hardware	Equivalence (Sanger et al., 2019), FlowAdapter (Pan et al., 2013)
	Domain-specific languages	[WPT]	SDN controller verification	Featherweight OpenFlow (Guha et al., 2013a,b), Coq (Bertot, 2008)
	First order logic, Logical connective	[WP]	Verification of security attributes in OpenFlow	MSAID (Li et al., 2019), FLOVER (Son et al., 2013)
ST, AL	Rule Decomposition, Graph-based model	[PT]	Rule decomposition, distribution and placement	Palette (Kanizo et al., 2013)
LC, FR	CAP (consistency, Availability, Partition) theorem (Gilbert and Lynch, 2002)	[WT]	Trade-off between policy enforcement and network connectivity	CAP for Networks (Panda et al., 2013)
ST, FR	Algorithmic Policies	[WPT]	Simplify SDN Programming	Maple (Voellmy et al., 2013)

[a]Modelled attributes: W — Wildcard; P — Priority; M — Multi-table; T — Topology

objects to define the scope of the statements. Predicate logic is a generic term of ***higher-order logic***.

These logics can describe functional relationships and statements about "for all" objects or about "for some" objects but their quantifiers vary. First-order logic quantifies over individuals of the domain of discourse. Higher-order logic is distinguished from first-order logic by additional quantifiers. As illustrated in Fig. 10, higher-order logic quantifies over sets or sets of sets while first-order logic has the quantifiers of non-nested sets.

Modal logic extends the classical proposition logic and predicate logic to allow the quantifiers to express modality, for example, "necessarily" and "possibly" (Emerson, 1990).

Compared to logical connectives and Boolean algebra, the aforementioned logics in Fig. 10 are high-level abstraction which indicates that they are better at expressing the connections among FTEs, for example, the priority and topology. However, the low-level FTE attributes such as wildcard and "goto table" actions are easily overlooked by these logics.

4.4.2. Mathematical logic

In mathematical logic, Boolean algebra is a branch of algebra in which the values of the variables are the truth values "true" and "false", usually denoted by 1 and 0, respectively. Boolean algebra is naturally suitable to represent SDN forwarding pipeline especially on the applications such as the equivalent forwarding set evaluation and traffic monitoring. In the former scenario, the question turns to the Boolean function on the match fields for the same action. In the latter

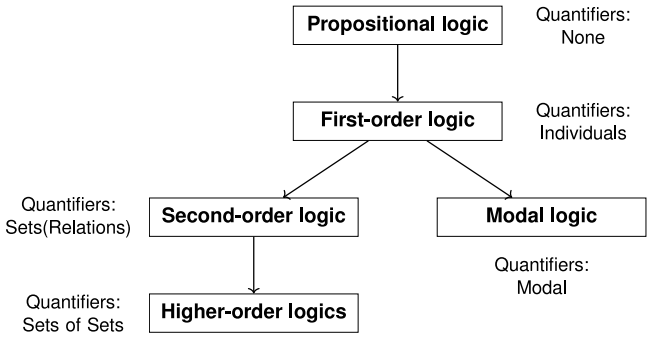


Fig. 10. Quantifiers of logic.

scenario, the question turns to find all supersets/subsets and retrieve their individual statistics for a given flowset.

To fit the Boolean algebra to a FTE, two challenges must be addressed: wildcard logical operations and deprioritisation. In traditional networking, wildcard algebra is only supported on lower bits (at the rear) of an IP address or MAC address to facilitate the longest prefix search, it may locate at any position in a FTE. Thus the Boolean algebra on FTE must allow the wildcard be positioned anywhere and accepted by all Boolean operations.

In this survey, "wildcard" stands for "wildcard mask" which was first coined by Cisco. It is a mask of bits that indicates which parts of IP addresses are available for examination. For example, a wildcard mask 0.0.0.3 represents that the last two bits are not available for examination. A packet with an IP address which ends with bit 0 or 1 will have the same examining result. A wildcard mask is continuous and always indicates the last certain number of bits are not available for examination. Thus a wildcard mask in a truth table has the following format: 1000011111****, 000111**, 110000* in which the number of asterisk is flexible while they always position at the rear of a Boolean value. However, in FTE, a more flexible format of wildcard is expected in a truth table, the wildcard (*) should be positioned anywhere (not only in the rear of a value), such as 10000**111****, 10000**0011, ***111000.

Another challenge is introduced by a FTE's unique attribute: "priority". Due to the prioritisation within a forwarding table, most FTEs' match fields cannot reveal their actual matching scope. After appropriate adaptations and extensions, Boolean algebra extends the existing SDN application into multiple flow tables across multiple switches.

Besides the intrinsic FTE attributes, wildcard and priority, Boolean algebra is also capable of representing another two attributes: multi-table and topology. The Boolean operations (for example, logical AND ($\wedge$) and OR ($\vee$)) between any two forwarding tables or even two switches can be converted to the operations between the FTEs in their respective tables or switches.

Besides Boolean algebra, other mathematical logical methods have also been examined to investigate the possibility of representing FTE and exploiting their applications. It is found that propositional logic and some lower-order logic are also good at expressing SDN forwarding pipeline. However, the abstraction in the process of FTE representation makes these approaches less applicable in practical scenarios where some low-level information is already lost. For example, the relative positions of multiple forwarding table entries and the connections among multiple tables are hard to be represented in these logics.

Among all the formal semantics, NetKAT represents single table very well and maintains most essential low-level information. However, the challenges on Boolean algebra exist for NetKAT as well, which means the similar adaptations and extensions on Boolean algebra are also required for NetKAT. Moreover, its initial design does not cover the multi-table attribute which further limits its applicable scope. For low-level representation, Boolean algebra is unlikely to reveal the whole picture of network-wide forwarding pipeline. However, it can be easily extended to support the latest SDN core forwarding attributes such as priority and multi-table.

Some research in Table 4 adopts the data structure based representation such as brick, tree and graph (Yu et al., 2011; Pan et al., 2013; Kang et al., 2013b; Kanizo et al., 2013; Moshref et al., 2013; Panda et al., 2013; Yan et al., 2014a; Giroire et al., 2014; Marsh, 2015; Kuan and Dimyati, 2009). They are not discussed in this survey because they usually aim to resolve a specific problem rather than providing a generic solution. For example, the RuleBricks (Williams and Jamjoom, 2013) provides high availability (HA) policies to OpenFlow forwarding pipeline. This survey intends to find a representation which is not particular to any application.

5. FTE representation in practice

Based on the analysis of FTE, various applications have been developed. These applications can be mainly categorised into two types from functionality perspective: FTE placement and Traffic monitoring. FTE placement mainly covers the following aspects: (i) FTE manipulation (Curtis et al., 2011; Gember et al., 2012; Pan et al., 2013; Tootoonchian and Ganjali, 2010), (ii) consistent updates over multiple switches (Afek et al., 2014; Canini et al., 2013; Katta et al., 2013; Perešíni et al., 2013; Yuan et al., 2014a), and (iii) placement optimisation (Nguyen et al., 2016). All these three placement applications rely on the correct conversion among the different forms of tables across multiple switches. The works on traffic monitoring include traffic estimation (Zhang et al., 2014), traffic anomaly detection (Zhang, 2013b), verification of forwarding tables (Zeng et al., 2014), etc.

5.1. Optimising FTE

Prior works on FTE's placement optimisation can be further divided into two categories: single switch flow table optimisation (Williams and Jamjoom, 2013; Yan et al., 2014a; Katta et al., 2016; Gupta et al., 2016; Yu et al., 2016; Wang et al., 2016) and network-wide flow table optimisation (Giroire et al., 2014; Kang et al., 2013b; Kanizo et al., 2013; Marsh, 2015; Moshref et al., 2013; Nguyen et al., 2014; Panda et al., 2013; Voellmy et al., 2013; Yu et al., 2011; Guo et al., 2018).

The optimisation of single table usually focuses on compression which heavily relies on the mechanism "merge" and " split". The new installed rules are firstly merged with each other and then split according to the structure and size a hardware forwarding table. Then the neighbouring or function-alike flow entries can be further merged to save table space.

Another optimisation approach is to deploy FTEs according to their different access frequencies, for example, deploying the most frequently accessed FTE in a high-speed cache and the relatively unpopular FTEs in a low-speed cache (Katta et al., 2016). In the scenario of FTE compression, a number of practical advances to increase the applicability of a hardware resilience via forwarding table compression algorithm and compression-aware routing have been discussed (Stephens et al., 2016). A method to verify whether redundant rules exist and to avoid them during incremental deployment are also proposed and verified (Wang et al., 2016).

A major drawback of this "merge" and " split" approach is that it fails to explore FTE's multi-table capability. Some researchers notice the benefit of multiple flow tables and they perform partitioning and compression by distributing forwarding rules into different tables according to their respective capabilities (Gupta et al., 2016). Network-wide optimisation is usually achieved by "combination" and "distribution". In a given interval, all requests from different applications will be combined as a single composite request and then reorganises and distributes the FTEs to the devices. During this process, the exact path specified by the original application might be changed. It enriches the freedom to adjust the FTE placement across all applications, however,

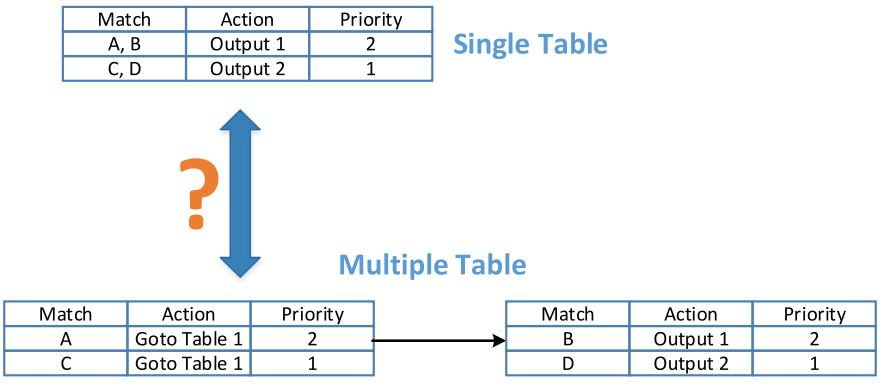


Fig. 11. Match fields distribution of OTN conversion.

the achievements of some higher-level goals such as traffic engineering cannot be guaranteed.

A core question for any deployment of multi-table FTE is related to assesing the equivalence between the table(s) to be provisioned and the table(s) deployed in the switches. Particular attention about multi-table evaluation and deployment should be paid to the research in Bosshart et al. (2013b), Pan et al. (2013) and Bakker et al. (2018). The former proposed a reconfigurable match tables (RMT) model which allows the forwarding plane to be modified by reorganising the tables (for example, adding new match fields and reconfiguring IP lookup tables) (Bosshart et al., 2013b). The latter proposed a middle layer to convert flow rules (FTEs) from the controller to a flow table pipeline of a hardware switch. In a hardware switch, the transformation from one single policy to a multi-table rule is usually required so that the rules can be fitted into different types of hardware (Pan et al., 2013). Both works in Bosshart et al. (2013b), Pan et al. (2013) and Guo et al. (2017) are concerned with the conversion between different types of forwarding tables.

However, it is found that the OTN (the conversion of a One-stage flow table into N-stage flow tables) proposed in Pan et al. (2013) is not always correct. Its core idea is to fill the corresponding match field value of a one-stage flow entry to multi-stage entries based on their containing match field types, as illustrated in Fig. 11.

In this conversion, the original single table (Table 1) has been converted into a multi-table structure (Tables 2 and 3). Take the first entry in Table 1 as an example, the original match fields {A, B} are decomposed and distributed into {A} in Table 2 and {B} in Table 3, respectively.

Closer analysis of the FTEs reveals that packets matching either {A, B} or {C, D} will behave the same while the packets with {A, D} or {C, B} will not. The packets with attributes {A, D} will NOT match any entries in the single table and will be dropped by default. However, in the converted multi-table, the same packets can match the first entry in Table 2 and then the second entry in Table 3 sequentially which means the matching packets will egress out of port 2. For the same packets, the forwarding behaviour will not be the same which means the single table is not equivalent to the multi-table. Thus, a systematic approach to evaluate the equivalence of any two given forwarding sets becomes a challenge for all forwarding table manipulations.

5.2. Traffic monitoring

FTEs can be used as a modelling tool to design network monitoring solutions, e.g. placing FTEs across a certain end-to-end path. The most widely adopted network traffic monitoring approaches are either based on packet sampling or sketch-based measurements. Sampling-based monitoring solutions are dominated by NetFlow (Hofstede et al., 2014) and sFlow (Phaal and Lavine, 2004). NetFlow is a feature on Cisco routers for collecting IP network traffic with predetermined rules. sFlow, short for "sampled flow" provides a means for collecting information of truncated packets together with the interface statistics. Since the set of FTEs in an SDN controller is like a "traffic map", the packet statistics related to a flow across a network can be interrogated and used as an estimation for traffic monitoring purposes (Yang et al., 2016b; Madanapalli et al., 2018).

Traffic monitoring with sFlow offers greater scalability (compared to NetFlow) and it reports information on network traffic (corresponding to OSI (Open Systems Interconnection) layer two to layer seven) in detail, but it consumes more resources (e.g., CPU, memory, and bandwidth) (Chakchai So-In and So-In, 2009; Bakker et al., 2019). A high sampling rate generates too much information (costly to store), while a lower sampling rate may result in heavy-hitter flows going undetected.

Compared to a sampling-based monitoring, a sketch-based approach can process millions of streams in a short time with low overheads. A sketch-based approach is a probabilistic summary of data streams within a compact data structure that builds forecast models on top of sketches which represent the past traffic patterns. A Sketch-based approach adopts an unique hash function and associates multi-dimensional tables to data streams for storing summarised data which requires customisation of existing switch ASIC. This is why most of the existing works are only verified by simulation and implemented on field-programmable gate array (FPGA) (Das et al., 2008; Yu et al., 2013).

To fit the aforementioned approaches into the realistic network monitoring scenario, ProgME (Programmable Network MEasurement) presents a framework to measure a flowset defined according to an application requirement (Yuan et al., 2011). A statistics query is processed by a query answering engine which maps a flowset to unique flows whose statistics can be retrieved directly from a hardware. Even though the scenarios of these traffic monitoring research vary, they share the same principle, i.e., to measure an arbitrary set of flows' traffic based on the ready statistics.

No matter what kind of approaches and frameworks are chosen, all the existing SDN traffic monitoring solutions use the same way to get the statistics. They install the application specified rules and then retrieve their statistics (Jose et al., 2011; Malboubi et al., 2014; Yu et al., 2013, 2014; Zhang, 2013). The benefit is that the controller can customise the FTE rules for flexible traffic monitoring, for example, the controller is able to update the FTE rules to monitor suspected malicious traffic dynamically. However, it is difficult to avoid the interference on the active traffic because the behaviour of all the packets matching these monitoring FTEs will be altered.

6. Conclusion

This paper reviews the recent literature on the representation and application of packet forwarding pipeline. Software Defined Networking (SDN) enables a network to be intelligently and centrally controlled

via forwarding table entries. The structure and functionality of FTE are more complicated than the traditional networking functions such as ACL and IP routing because of its generalisation. We propose a new taxonomy that is based on the FTE typology and the OpenFlow attributes each typology models to categorise and study the different approaches to FTE representation. We apply this taxonomy to classify different projects and research works reported in the literature. Furthermore, we relate the FTE representation to use cases in practice and analyse how the representations facilitate research in SDN.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research is partly supported by Victoria's Huawei NZ Research Programme, Software-Defined Green Internet of Things project (E2881) and Victoria Doctoral Scholarship.

References

Afek, Y., Bremler-Barr, A., Schiff, L., 2014. Ranges and cross-entrance consistency with openflow. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, pp. 233–234.

Al-Shaer, E., Al-Haj, S., 2010. FlowChecker: Configuration analysis and verification of federated openflow infrastructures. In: Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration. ACM, pp. 37–44.

Alsaeedi, M., Mohamad, M.M., Al-Roubaiey, A.A., 2019. Toward adaptive and scalable OpenFlow-SDN flow control: A survey. IEEE Access 7, 107346–107379.

Anderson, C.J., Foster, N., Guha, A., Jeannin, J.-B., Kozen, D., Schlesinger, C., Walker, D., 2014. NetKAT: Semantic foundations for networks. ACM SIGPLAN Not. 49 (1), 113–126.

Bakker, J.N., Ng, B., Seah, W.K., 2018. Can machine learning techniques be effectively used in real networks against ddos attacks?. In: 2018 27th International Conference on Computer Communication and Networks. ICCCN, IEEE, pp. 1–6.

Bakker, J., Ng, B., Seah, W.K., Pekar, A., 2019. Traffic classification with machine learning in a live network. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management. IM, IEEE, pp. 488–493.

Barkley, J., 1997. Comparing simple role based access control models and access control lists. In: Proceedings of the Second ACM Workshop on Role-Based Access Control. ACM, pp. 127–132.

Ben Azzouz, L., Jamai, I., 2019. SDN, slicing, and NFV paradigms for a smart home: A comprehensive survey. Trans. Emerg. Telecommun. Technol. 30 (10), e3744.

Benson, T., Akella, A., Maltz, D.A., 2009. Mining policies from enterprise network configuration. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. IMC '09, ACM, New York, NY, USA, pp. 136–142.

Bertot, Y., 2008. A short presentation of coq. In: TPHOLs, Vol. 8. Springer, pp. 12–16.

Bifulco, R., Schneider, F., 2013. OpenFlow rules interactions: definition and detection. In: SDN for Future Networks and Services. SDN4FNS, IEEE, pp. 1–6.

Bosshart, P., Gibb, G., Kim, H.-S., Varghese, G., McKeown, N., Izzard, M., Mujica, F., Horowitz, M., 2013a. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM. SIGCOMM '13, ACM, New York, NY, USA, pp. 99–110.

Bosshart, P., Gibb, G., Kim, H.-S., Varghese, G., McKeown, N., Izzard, M., Mujica, F., Horowitz, M., 2013b. Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn. ACM SIGCOMM Comput. Commun. Rev. 43 (4), 99–110.

Canini, M., Kuznetsov, P., Levin, D., Schmid, S., 2013. Software transactional networking: Concurrent and consistent policy composition. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 1–6.

Canini, M., Venzano, D., Peresini, P., Kostic, D., Rexford, J., et al., 2012. A NICE way to test openflow applications. In: NSDI, Vol. 12. pp. 127–140.

Chakchai So-In, So-In, C., 2009. A survey of network traffic monitoring and analysis tools. In: Cse 576m computer system analysis project, Washington University in St. Louis. Citeseer, Washington University in St. Louis.

Clarke, E.M., 2012. Lecture 1: Propositional logic. https://www.cs.cmu.edu/~emc/15414-f12/lecture/propositional_logic.pdf.

Codd, E.F., 1972. Relational Completeness of Data Base Sublanguages. IBM Corporation.

Curtis, A.R., Mogul, J.C., Tourrilhes, J., Yalagandula, P., Sharma, P., Banerjee, S., 2011. DevoFlow: Scaling flow management for high-performance networks. ACM SIGCOMM Comput. Commun. Rev. 41 (4), 254–265.

Dargahi, T., Caponi, A., Ambrosin, M., Bianchi, G., Conti, M., 2017. A survey on the security of stateful SDN data planes. IEEE Commun. Surv. Tutor. 19 (3), 1701–1725.

Das, A., Nguyen, D., Zambreno, J., Memik, G., Choudhary, A., 2008. An FPGA-based network intrusion detection architecture. IEEE Trans. Inf. Forensics Secur. 3 (1), 118–132.

Emerson, E.A., 1990. Temporal and modal logic.. In: Handbook of Theoretical Computer Science, Vol. B. In: Formal Models and Sematics (B), vol. 995, (1072), p. 5.

Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A., 2011. Network configuration protocol (NETCONF). RFC 6241.

Farris, I., Taleb, T., Khettab, Y., Song, J., 2018. A survey on emerging SDN and NFV security mechanisms for IoT systems. IEEE Commun. Surv. Tutor. 21 (1), 812–837.

Feamster, N., Rexford, J., Zegura, E., 2013. The road to SDN. Queue 11 (12), 20.

Ferraiolo, D., Kuhn, D.R., Chandramouli, R., 2003. Role-Based Access Control. Artech House.

Foster, N., Harrison, R., Freedman, M.J., Monsanto, C., Rexford, J., Story, A., Walker, D., 2011. Frenetic: A network programming language. ACM SIGPLAN Not. 46 (9), 279–291.

Foster, N., Kozen, D., Milano, M., Silva, A., Thompson, L., 2015. A coalgebraic decision procedure for NetKAT. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL. pp. 343–355.

Gember, A., Prabhu, P., Ghadiyali, Z., Akella, A., 2012. Toward software-defined middlebox networking. In: Proceedings of the 11th ACM Workshop on Hot Topics in Networks. ACM, pp. 7–12.

Gilbert, S., Lynch, N., 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. Acm Sigact News 33 (2), 51–59.

Giroire, F., Moulierac, J., Phan, T.K., 2014. Optimizing rule placement in software-defined networks for energy-aware routing. In: Proceedings of the IEEE Global Communications Conference. GlobeCom. Austin, TX, USA. pp. 1–7.

Goransson, P., Black, C., 2014. Software Defined Networks: A Comprehensive Approach. Elsevier.

Gordon, C.S., 2018. Polymorphic iterable sequential effect systems. arXiv preprint arXiv:1808.02010.

Guha, A., Reitblatt, M., Foster, N., 2013a. Formal foundations for software defined networks. Open Net Summ..

Guha, A., Reitblatt, M., Foster, N., 2013b. Machine-verified network controllers. ACM SIGPLAN Not. 48 (6), 483–494.

Guo, Z., Liu, R., Xu, Y., Gushchin, A., Walid, A., Chao, H.J., 2017. STAR: Preventing flow-table overflow in software-defined networks. Comput. Netw. 125, 15–25.

Guo, Z., Xu, Y., Liu, R., Gushchin, A., Chen, K.-y., Walid, A., Chao, H.J., 2018. Balancing flow table occupancy and link utilization in software-defined networks. Future Gener. Comput. Syst. 89, 213–223.

Gupta, A., MacDavid, R., Birkner, R., Canini, M., Feamster, N., Rexford, J., Vanbever, L., 2016. An industrial-scale software defined internet exchange point. In: 13th USENIX Symposium on Networked Systems Design and Implementation. NSDI 16, USENIX Association, Santa Clara, CA, pp. 1–14.

Gutz, S., Story, A., Schlesinger, C., Foster, N., 2012. Splendid isolation: A slice abstraction for software-defined networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. ACM, pp. 79–84.

Habib, B., Khurshid, F., Dar, A.H., Shah, Z., 2019. DDoS mitigation in eucalyptus cloud platform using snort and packet filtering — IP-tables. In: 2019 4th International Conference on Information Systems and Computer Networks. ISCON. pp. 546–550.

Haleplidis, E., Salim, J.H., Denazis, S., Koufopavlou, O., 2015. Towards a network abstraction model for SDN. J. Netw. Syst. Manage. 23 (2), 309–327.

Hao, L., Ng, B., 2018. Using genetic algorithms based on neighbor list mechanism to reduce handover latency for IEEE 802.11 WLAN. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 235–236.

Hao, L., Ng, B., 2019. Self-healing solutions for wi-fi networks to provide seamless handover. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management. IM, IEEE, pp. 639–642.

Hao, L., Ng, B., Qu, Y., 2017. Dynamic optimization of neighbor list to reduce changeover latency for wi-fi networks. In: Proceedings of the 2017 International Conference on Telecommunications and Communication Engineering. pp. 20–24.

Hofstede, R., Celeda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., Pras, A., 2014. Flow monitoring explained: From packet capture to data analysis with netflow and IPFIX. IEEE Commun. Surv. Tutor. 16 (4), 2037–2064.

Hossein, A., Watts, M., Ahmadi, K., 2019. An overview of multi-controller architecture in software-defined networking. In: CITRENZ Conference (2019). Nelson, NZ. pp. 1–7.

Hsueh, S.-W., Lin, T.-Y., Lei, W.-I., Ngai, C.-L.P., Sheng, Y.-H., Wu, Y.-S., 2018. Semantic failover in software-defined networking. In: 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing. PRDC, IEEE, pp. 299–308.

Huang, N.-F., Zhao, S.-M., 1999. A novel IP-routing lookup scheme and hardware architecture for multigigabit switching routers. IEEE J. Sel. Areas Commun. 17 (6), 1093–1104.

Jindal, A., Aujla, G.S., Kumar, N., 2019. SURVIVOR: A blockchain based edge-as-a-service framework for secure energy trading in SDN-enabled vehicle-to-grid environment. Comput. Netw. 153, 36–48.

Jose, L., Yu, M., Rexford, J., 2011. Online measurement of large traffic aggregates on commodity switches. In: Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. Hot-ICE. pp. 1–13.

Kang, N., Liu, Z., Rexford, J., Walker, D., 2013a. Optimizing the "one big switch" abstraction in software-defined networks. In: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies. CoNEXT '13, ACM, New York, NY, USA, pp. 13–24.

Kang, N., Liu, Z., Rexford, J., Walker, D., 2013b. Optimizing the one big switch abstraction in software-defined networks. In: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies. ACM, pp. 13–24.

Kanizo, Y., Hay, D., Keslassy, I., 2013. Palette: Distributing tables in software-defined networks. In: INFOCOM, 2013 Proceedings IEEE. IEEE, pp. 545–549.

Kantor, M., Biernacka, E., Boryło, P., Domżał, J., Jurkiewicz, P., Stypiński, M., Wójcik, R., 2019. A survey on multi-layer IP and optical Software-Defined Networks. Comput. Netw. 162, 106844.

Kao, Y.-C., Liu, J.-C., Wang, Y.-H., Chu, Y.-H., Tsai, S.-C., Lin, Y.-B., 2019. Automatic blocking mechanism for information security with SDN. J. Internet Serv. Inf. Secur. 9 (1), 60–73.

Katta, N., Alipourfard, O., Rexford, J., Walker, D., 2014. Infinite cacheflow in software-defined networks. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, pp. 175–180.

Katta, N., Alipourfard, O., Rexford, J., Walker, D., 2016. Cacheflow: Dependency-aware rule-caching for software-defined networks. In: Proc. ACM Symposium on SDN Research. SOSR. pp. 1–12.

Katta, N.P., Rexford, J., Walker, D., 2013. Incremental consistent updates. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 49–54.

Kozen, D., 1997. Kleene algebra with tests. ACM Trans. Programm. Lang. Syst. 19 (3), 427–443.

Kuan, C., Dimyati, K., 2006. Analysis of collision probabilities for saturated IEEE 802.11 MAC protocol. Electron. Lett. 42 (19), 1.

Kuan, C., Dimyati, K., 2009. Finite time-horizon Markov model for IEEE 802.11 e. J. Zhejiang Univ.-SCI. A 10 (10), 1383–1388.

Kyung, Y., Park, J., 2019. Prioritized admission control with load distribution over multiple controllers for scalable SDN-based mobile networks. Wirel. Netw. 25 (6), 2963–2976.

Lampson, B.W., 1974. Protection. SIGOPS Oper. Syst. Rev. 8 (1), 18–24.

Levin, D., Wundsam, A., Heller, B., Handigol, N., Feldmann, A., 2012. Logically centralized?: State distribution trade-offs in software defined networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. HotSDN '12, ACM, New York, NY, USA, pp. 1–6.

Li, W., Meng, W., Kwok, L.F., 2016. A survey on openflow-based software defined networks: Security challenges and countermeasures. J. Netw. Comput. Appl. 68, 126–139.

Li, Y., Wang, Z., Yao, J., Yin, X., Shi, X., Wu, J., Zhang, H., 2019. MSAID: Automated detection of interference in multiple SDN applications. Comput. Netw. 153, 49–62.

Li, Y., Yin, X., Wang, Z., Yao, J., Shi, X., Wu, J., Zhang, H., Wang, Q., 2018. A survey on network verification and testing with formal methods: Approaches and challenges. IEEE Commun. Surv. Tutor. 21 (1), 940–969.

Lin, Y.-D., Lai, Y.-K., Tsou, Y.-L., Lai, Y.-C., Liou, E.-C., Chiang, Y., 2019. Generic validation criteria and methodologies for SDN applications. IEEE Syst. J..

Liu, Y., Zhao, B., Zhao, P., Fan, P., Liu, H., 2019. A survey: Typical security issues of software-defined networking. China Commun. 16 (7), 13–31.

Lu, J., Zhang, Z., Hu, T., Yi, P., Lan, J., 2019. A survey of controller placement problem in software-defined networking. IEEE Access 7, 24290–24307.

Madanapalli, S.C., Lyu, M., Kumar, H., Gharakheili, H.H., Sivaraman, V., 2018. Real-time detection, isolation and monitoring of elephant flows using commodity SDN system. In: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. IEEE, pp. 1–5.

Malboubi, M., Wang, L., Chuah, C.N., Sharma, P., 2014. Intelligent SDN based traffic (de)Aggregation and Measurement Paradigm (iSTAMP). In: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications. pp. 934–942.

Marsh, C., 2015. A generalized algorithm for flow table optimization. http://www.crmarsh.com/static/pdf/Charles_Marsh_SDN.pdf.

Masoudi, R., Ghaffari, A., 2016. Software defined networks: A survey. J. Netw. Comput. Appl. 67, 1–25.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. Openflow: Enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38 (2), 69–74.

Michaelis, J., Diekmann, C., 2016. Routing. Arch. Formal Proofs URL http://isa-afp.org/entries/Routing.shtml.

Monsanto, C., Foster, N., Harrison, R., Walker, D., 2012. A compiler and run-time system for network programming languages. ACM SIGPLAN Not. 47 (1), 217–230.

Moshref, M., Yu, M., Sharma, A.B., Govindan, R., 2013. Scalable rule management for data centers. In: NSDI. pp. 157–170.

Mycroft, A., Orchard, D., Petricek, T., 2016. Effect systems revisited—control-flow algebra and semantics. In: Semantics, Logics, and Calculi. Springer, pp. 1–32.

Ng, B., Tan, Y., Roger, Y., 2013. Improved utilization for joint HCCA–EDCA access in IEEE 802.11 e WLANs. Optim. Lett. 7 (8), 1711–1724.

Nguyen, X.N., Saucez, D., Barakat, C., Turletti, T., 2014. Optimizing rules placement in OpenFlow networks: trading routing for better efficiency. In: ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. HotSDN 2014. pp. 127–132.

Nguyen, X.N., Saucez, D., Barakat, C., Turletti, T., 2016. Rules placement problem in openflow networks: A survey. IEEE Commun. Surv. Tutor. 18 (2), 1273–1286.

Nunes, B., Mendonca, M., Xuan-Nam Nguyen, O., K., T., 2014. A survey of software-defined networking: Past, present, and future of programmable networks. Commun. Surv. Amp; Tutor. IEEE 16 (3), 1617–1634.

Omar, N., Samsudin, A.T., 2019. Hybrid software-defined network monitoring. In: Internet and Distributed Computing Systems: 12th International Conference, IDCS 2019, Naples, Italy, October 10–12, 2019, Proceedings, Vol. 11874. Springer Nature, p. 234.

ONF, 2011. Specification, openflow switch, version 1.1.0. http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf.

ONF, 2014. Specification, openflow switch, version 1.5.0. https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-switch-v1.5.0.pdf.

Pan, H., Guan, H., Liu, J., Ding, W., Lin, C., Xie, G., 2013. The FlowAdapter: Enable flexible multi-table processing on legacy hardware. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 85–90.

Panda, A., Scott, C., Ghodsi, A., Koponen, T., Shenker, S., 2013. Cap for networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 91–96.

Perešíni, P., Kuzniar, M., Vasić, N., Canini, M., Kostiū, D., 2013. OF. CPP: Consistent packet processing for OpenFlow. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 97–102.

Pfaff, B., Heller, B., Talayco, D., Erickson, D., Gibb, G., Appenzeller, G., Tourrilhes, J., Pettit, J., Yap, K., Casado, M., et al., 2009. Openflow switch specification.

Pfaff, B., Pettit, J., Koponen, T., Jackson, E.J., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., et al., 2015. The design and implementation of open vswitch. In: NSDI, Vol. 15. pp. 117–130.

Phaal, P., Lavine, M., 2004. Sflow version 5. http://sflow.org/sflow_version_5.txt.

Pozo, S., Ceballos, R., Gasca, R.M., 2008. Afpl, an abstract language model for firewall acls. In: International Conference on Computational Science and Its Applications. Springer, pp. 468–483.

Priya, I.D., Silas, S., 2019. A survey on research challenges and applications in empowering the SDN-based internet of things. In: Advances in Big Data and Cloud Computing. Springer, pp. 457–467.

Qiu, K., Yuan, J., Zhao, J., Wang, X., Secci, S., Fu, X., 2019. Fastrule: Efficient flow entry updates for TCAM-based openflow switches. IEEE J. Sel. Areas Commun. 37 (3), 484–498.

Rekhter, Y., Li, T., Hares, S., 1994. A border gateway protocol 4 (BGP-4). Internet Engineering Task Force, RFC 1654.

Riener, H., Testa, E., Haaswijk, W., Mishchenko, A., Amarù, L., De Micheli, G., Soeken, M., 2019. Scalable generic logic synthesis: One approach to rule them all. In: Proceedings of the 56th Annual Design Automation Conference 2019. ACM, p. 70.

Rojas, E., Doriguzzi-Corin, R., Tamurejo, S., Beato, A., Schwabe, A., Phemius, K., Guerrero, C., 2018. Are we ready to drive software-defined networks? A comprehensive survey on management tools and techniques. ACM Comput. Surv. 51 (2), 27.

Salman, O., Elhajj, I., Chehab, A., Kayssi, A., 2018. IoT survey: An SDN and fog computing perspective. Comput. Netw. 143, 221–246.

Sanger, R., Luckie, M., Nelson, R., 2019. Identifying equivalent SDN forwarding behaviour. In: Proceedings of the 2019 ACM Symposium on SDN Research. ACM, pp. 127–139.

Saraswat, S., Agarwal, V., Gupta, H.P., Mishra, R., Gupta, A., Dutta, T., 2019. Challenges and solutions in Software Defined Networking: A survey. J. Netw. Comput. Appl. 141, 23–58.

Shin, M.-K., Kang, M., Choi, J.-Y., Nam, K.-H., 2013. Formal specification for software-defined networks (SDN). https://tools.ietf.org/html/draft-shin-sdn-formal-specification-01.

Shin, M.-K., Nam, K.-H., Kim, H.-J., 2012. Software-defined networking (SDN): A reference architecture and open APIs. In: ICT Convergence (ICTC), 2012 International Conference on. IEEE, pp. 360–361.

Skalka, C., Ring, J., Darias, D., Kwon, M., Gupta, S., Diller, K., Smolka, S., Foster, N., 2019. Proof-carrying network code. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 1115–1129.

Smith, M., Dvorkin, M., Laribi, Y., Pandey, V., Garg, P., Weidenbacher, N., 2014. OpFlex control protocol. IETF, Apr.

Son, S., Shin, S., Yegneswaran, V., Porras, P., Gu, G., 2013. Model checking invariant security properties in OpenFlow. In: Communications (ICC), 2013 IEEE International Conference on. IEEE, pp. 1974–1979.

Souri, A., Norouzi, M., Asghari, P., Rahmani, A.M., Emadi, G., 2019. A systematic literature review on formal verification of software-defined networks. Trans. Emerg. Telecommun. Technol..

Stephens, B., Cox, A.L., Rixner, S., 2016. Scalable multi-failure fast failover via forwarding table compression. In: Proceedings of the Proceedings of the Symposium on SDN Research. SOSR. pp. 1–12.

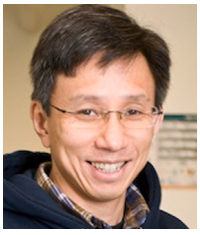
Su, J., Wang, W., Liu, C., 2019. A survey of control consistency in Software-Defined Networking. CCF Trans. Netw. 1–16.

Sultana, N., Chilamkurti, N., Peng, W., Alhadad, R., 2019. Survey on SDN based network intrusion detection system using machine learning approaches. Peer-to-Peer Netw. Appl. 12 (2), 493–501.

Tootoonchian, A., Ganjali, Y., 2010. HyperFlow: A distributed control plane for OpenFlow. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking. USENIX Association, pp. 1–3.
Ullman, J., 1994. Chapter 14 predicate logic. http://infolab.stanford.edu/~ullman/focs/ch14.pdf.
Vardi, G., Kupferman, O., 2019. Flow logic. Log. Methods Comput. Sci. 15.
Voellmy, A., Wang, J., Yang, Y.R., Ford, B., Hudak, P., 2013. Maple: Simplifying SDN programming using algorithmic policies. ACM SIGCOMM Comput. Commun. Rev. 43 (4), 87–98.
Wang, Y., Bi, J., Lin, P., Lin, Y., Zhang, K., 2016. SDI: a multi-domain SDN mechanism for fine-grained inter-domain routing. Ann. Telecommun. 1–13.
Wang, A., Moarref, S., Loo, B.T., Topcu, U., Scedrov, A., 2013. Automated synthesis of reactive controllers for software-defined networks. In: Network Protocols (ICNP), 2013 21st IEEE International Conference on. IEEE, pp. 1–6.
Whitesitt, J.E., 1995. Boolean Algebra and Its Applications. Courier Corporation.
Williams, D., Jamjoom, H., 2013. Cementing high availability in OpenFlow with RuleBricks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 139–144.
Wong, M.K., Gajjar, Y.V., Kumar, R., 2010. Efficient ACL lookup algorithms. Google Patents. US Patent 7, 808, 929.
Xie, J., Yu, F.R., Huang, T., Xie, R., Liu, J., Wang, C., Liu, Y., 2018. A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. IEEE Commun. Surv. Tutor. 21 (1), 393–430.
Yan, B., Xu, Y., Chao, H.J., 2018. Bigmac: Reactive network-wide policy caching for SDN policy enforcement. IEEE J. Sel. Areas Commun. 36 (12), 2675–2687.
Yan, B., Xu, Y., Xing, H., Xi, K., Chao, H.J., 2014a. CAB: a reactive wildcard rule caching system for software-defined networks. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, pp. 163–168.
Yang, L., Ng, B., Seah, W.K.G., 2016a. Heavy hitter detection and identification in software defined networking. In: 2016 25th International Conference on Computer Communication and Networks. ICCCN. pp. 1–10.
Yang, L., Ng, B., Seah, W.K., 2016b. Heavy hitter detection and identification in software defined networking. In: 2016 25th International Conference on Computer Communication and Networks. ICCCN, IEEE, pp. 1–10.
Yang, L., Ng, B., Seah, W.K.G., Groves, L., 2017a. Equivalent forwarding set evaluation in software defined networking. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management. IM. pp. 576–579.
Yang, L., Ng, B., Seah, W.K., Groves, L., 2017b. Deterministic confidence interval estimation of networking traffic in SDN. In: 2017 IEEE 42nd Conference on Local Computer Networks. LCN, IEEE, pp. 120–127.
Yang, R., Yu, F.R., Si, P., Yang, Z., Zhang, Y., 2019. Integrated blockchain and edge computing systems: A survey, some research issues and challenges. IEEE Commun. Surv. Tutor. 21 (2), 1508–1532.
Yu, M., Jose, L., Miao, R., 2013. Software defined traffic measurement with opensketch. In: Presented As Part of the 10th USENIX Symposium on Networked Systems Design and Implementation. NSDI 13. pp. 29–42.
Yu, Y., Li, X., Leng, X., Song, L., Bu, K., Chen, Y., Yang, J., Zhang, L., Cheng, K., Xiao, X., 2018. Fault management in software-defined networking: A survey. IEEE Commun. Surv. Tutor. 21 (1), 349–392.
Yu, C., Lumezanu, C., Madhyastha, H.V., Jiang, G., 2016. Characterizing rule compression mechanisms in software-defined networks. In: International Conference on Passive and Active Network Measurement. Springer, pp. 302–315.
Yu, Y., Qian, C., Li, X., 2014. Distributed and collaborative traffic monitoring in software defined networks. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. HotSDN '14, ACM, New York, NY, USA, pp. 85–90.
Yu, M., Rexford, J., Freedman, M.J., Wang, J., 2011. Scalable flow-based networking with DIFANE. ACM SIGCOMM Comput. Commun. Rev. 41 (4), 351–362.
Yuan, L., Chuah, C.-N., Mohapatra, P., 2011. ProgME: Towards programmable network measurement. IEEE/ACM Trans. Netw. 19 (1), 115–128.
Yuan, Y., Ivančić, F., Lumezanu, C., Zhang, S., Gupta, A., 2014a. Generating consistent updates for software-defined network configurations. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, pp. 221–222.
Zeng, H., Zhang, S., Ye, F., Jeyakumar, V., Ju, M., Liu, J., McKeown, N., Vahdat, A., 2014. Libra: Divide and conquer to verify forwarding tables in huge networks. In: Proceedings of NSDI, Vol. 14. pp. 87–99.
Zhang, Y., 2013. An adaptive flow counting method for anomaly detection in SDN. In: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies. CoNEXT '13, ACM, New York, NY, USA, pp. 25–30.
Zhang, Y., 2013b. An adaptive flow counting method for anomaly detection in sdn. In: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies. ACM, pp. 25–30.
Zhang, Y., Cui, L., Wang, W., Zhang, Y., 2018a. A survey on software defined networking with multiple controllers. J. Netw. Comput. Appl. 103, 101–118.
Zhang, H., Lumezanu, C., Rhee, J., Arora, N., Xu, Q., Jiang, G., 2014. Enabling layer 2 pathlet tracing through context encoding in software-defined networking. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. HotSDN. pp. 169–174.
Zhang, P., Zhang, C., Hu, C., 2018b. Fast data plane testing for software-defined networks with rulechecker. IEEE/ACM Trans. Netw. 27 (1), 173–186.
Zhou, Z., Benson, T.A., 2019. Composing SDN controller enhancements with mozart. In: Proceedings of the ACM Symposium on Cloud Computing. ACM, pp. 351–363.

Liang Yang received the Dr.Eng. degree from Victoria University of Wellington, New Zealand, in 2018. He is a senior software engineer and data scientist. Prior to this, he has worked for more than 15 years in commercial research laboratories such as Motorola, Ericsson and IBM. His latest research interests include cloud computing, networking virtualization, data visualisation and data analysis.

Bryan Ng completed his Ph.D. (2010) in the area of communication and networking. He held teaching & research positions in Malaysia and France in addition to attachments to commercial research laboratories Intel, Motorola, Panasonic and Orange Labs. His research interests include performance analysis of communication networks, modelling networking protocols and software defined networking.



Winston K.G. Seah received the Dr.Eng. degree from Kyoto University, Kyoto, Japan, in 1997. He is currently Professor of Network Engineering in the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. Prior to this, he has worked for more than 16 years in mission-oriented industrial research, taking ideas from theory to prototypes, most recently, as a Senior Scientist in the Institute for Infocomm Research, Singapore. His latest research interests include Internet of Things, wireless sensor networks powered by ambient energy harvesting, wireless multi-hop networks, software defined networking, and 5G access protocols for machine-type communications.

Lindsay Groves is currently Associate Professor in the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. His current research is concerned with formal software development, with emphasis on structuring formal specifications, tool support for the refinement calculus, and techniques for combining and adapting formal derivations. He also has more general interests in software engineering, including program visualisation, program understanding, program maintenance/evolution and safety-critical systems.

Deepak Kumar Singh received the Dr.Eng. degree from Victoria University of Wellington, New Zealand, in 2019. He is currently working as a Research Assistant in Victoria University of Wellington, New Zealand. His research focuses on modelling of Software-Defined Network, data modelling and recommendation system.