Deterministic Confidence Interval Estimation of Networking Traffic in SDN

Liang Yang Bryan Ng Winston K.G. Seah Lindsay Groves
School of Engineering and Computer Science,
Victoria University of Wellington, New Zealand
{liang.yang, bryan.ng, winston.seah, lindsay}@ecs.vuw.ac.nz

***Abstract*—Software Defined Networking (SDN) enables a centralised entity - *the controller* - to monitor the network's status by collecting traffic statistics such as packets, bytes, etc. Each statistic is associated with a forwarding table entry (FTE) in a switch whose structure and format is specified by the OpenFlow standard (de-facto SDN standard). For a flow with a FTE, its statistic is easily acquired by an inquiry from a controller to the switch on this flow's corresponding FTE. If a flow has no matching FTE, its statistic is not known until a new FTE is installed for the purpose of monitoring it. However, the time to install these FTEs and the potential conflicts between them and the existing FTEs jeopardise the feasibility of this approach. To avoid these drawbacks, this paper proposes a traffic estimation approach based on the existing FTE's statistics. With the help of boolean algebra, the deterministic confidence interval of any given flowset can be estimated. This approach avoids the FTE installation time and also saves the FTE storage space.**

I. Introduction

Software Defined Networking (SDN) empowers a central entity, the controller, to manage packet forwarding and monitor networking traffic on all switches in a network. An SDN-enabled switch maintains a forwarding table in which each entry matches all the incoming packets to determine their next hops and also records their statistics. Here the flow is a sequence of packets sent from a particular source to a particular unicast, anycast or multicast destination. However, not all flows follow the aforementioned standard flow definition. Flows can belong to an arbitrary set of flows which share some common characteristics; such an arbitrary set of flows is also called a "flowset" [1].

If a flowset matches an existing FTE, its statistic can be easily retrieved from the corresponding FTE's associated statistic in a switch. However, if no FTE for a given flowset exists, the controller has to install a new FTE to monitor and collect this flowset's traffic. There are three drawbacks for this solution: i) extra delay is introduced into the traffic monitoring process due to the time needed to install the new FTE; ii) extra space is occupied by the monitoring FTEs; and iii) forwarding behaviour will be altered if conflicts happen between the newly installed FTE and the existing FTEs.

A new installed FTE cannot always guarantee that a controller retrieves the required statistic. Two conditions must be met by the monitoring-only FTE: i) it must have the highest priority to match against all incoming packets; and ii) it must have the lowest priority to avoid having any impact on the forwarding behaviour of incoming packets. These two contradictory conditions cannot be achieved on a single-table switch which is still popular due to backward compatibility for legacy hardware. For a multi-table switch, it can be achieved by reserving the first table (Table 0) for these monitoring FTEs if the match fields of these new installed FTEs are independent [2].

To eliminate these constraints, we proposed a DEterministic Confidence Interval eStimatION (DECISION) solution based on the relation between a given flowset and the existing FTEs. In our solution, there will be no new FTE installed into any switch. The deterministic confidence traffic interval for a given flowset is estimated by FTEs already installed in a switch and their associated statistics.

II. Definition

A traffic interval is a set of data with the property that any datum lies between two values in the set is also included in the set. For example, the set of all data x satisfying $0 \leq x \leq 1000$ is an interval which contains 0 and 1000 as well as all values between them. Here the traffic might be packet count, number of bytes or any units which reflect the amount of traffic in a flow. In this paper, we assume the statistics collecting from all flow tables use the same unit (packet counter or bytes). For the estimated traffic interval, the same unit will be adopted. To simplify the expression of traffic interval, their units will be omitted in the following description.

A traffic interval between a and b, including a and b, is denoted as $[a, b]$. If both a and b are finite and real numbers, the interval $[a, b]$ is bounded. In this case, a and b are lower bound and upper bound of this interval, respectively. For any given flowset, the traffic interval is always bounded because it must fall between 0 (no traffic at all) and the total traffic in a device. Thus, the problem of traffic interval estimation turns into finding the lower bound and upper bound for a given flowset. The absolute difference between these two bounds is also called *length* of an interval. The smaller the length of an interval is, the more precise an estimation becomes. Thus the best estimated interval must be composed by the greatest lower bound as well as the least upper bound.

Definition 1 (Deterministic Confidence Interval (DCI) [3])**.** *A confidence interval gives an estimated range of values so defined that there is a specified probability that the value of a parameter lies within it. Usually the estimated range is*

calculated from a given set of sample data. If the possibility reaches 100%, it becomes a deterministic confidence interval (DCI), which means the parameter always lies within the estimated range.

Definition 2 (Optimal DCI (ODCI))**.** *An optimal DCI (denoted by $\mathcal{O}$) is a DCI such that there does not exist any other DCI which is a subset of $\mathcal{O}$: $\nexists D \in \mathbb{D}, D \subset \mathcal{O}$ where $\mathbb{D}$ represents the full set of DCI for a given estimation. Thus an Optimal Deterministic Confidence Interval (ODCI) is bounded by the greatest lower bound (infimum) and the least upper bound (supremum): $\mathcal{O} = [lo, ro] : \nexists D = [lv, rv] \in \mathbb{D}, lo < lv, ro > rv$. They are denoted as the lower bound of ODCI (LODCI) and the upper bound of ODCI (UODCI), respectively.*

With the definition of ODCI, our proposed DECISION approach turns into the problem of finding ODCI for a given flowset.

III. Problem Description

A flowset is an arbitrary set of flows defined by the specified common fields (SCF), for example, a flowset f whose medium access control (MAC) address equals to "00:11:22:33:44:55" ($SCF_f = \{MAC = 00{:}11{:}22{:}33{:}44{:}55\}$) or a combination of a virtual LAN (VLAN) ID 1 and a Multiprotocol Label Switching (MPLS) label 101 ($SCF_f = \{VLAN = 1, MPLS = 101\}$). In this paper, SCF and a FTE's match fields are expressed in a form of boolean function, i.e., $SCF = g(a_1, a_2, \ldots, a_n)$ where a_i represents the attributes of a flowset which has been specified in latest OpenFlow standard [4] and function g represents a boolean logical operation (e.g., "AND($\wedge$)", "OR($\vee$))", "NOT($\neg$)", etc.) on these attributes. With the introduction of SCF, an ODCI estimation problem can be addressed with the help of set relation between SCF and the match fields of a forwarding set. The formal definition of this problem is expressed as follows:

Problem 1 (**Optimal DCI Estimation**)**.** *Given a forwarding set F and a flowset f which is defined by its specific common fields SCF_f, the ODCI estimation problem is to identify the traffic interval of f (denoted as T_f) whose length is as small as possible. Here F is composed of forwarding table entries in the form of single-table or multi-table, each entry is associated with a statistic. Specifically, the traffic interval is composed of lower bound l and upper bound u, i.e., $T_f = [l, u]$. Thus the ODCI estimation problem turns into the maximisation of l and the minimisation of u.*

For a switch, all the FTEs, routing tables, access control lists (ACL) and high level polices determine the forwarding behaviour of all incoming packets, these various types of forwarding entries are called *forwarding set*. In SDN, a forwarding set is interpreted in the form of a single table or chained multiple tables to enable packets processing in a more sophisticated way. The OpenFlow standard specifies the structure of a forwarding table which is illustrated in the top half of Fig. 1 [4]. The bottom half of Fig. 1 shows a

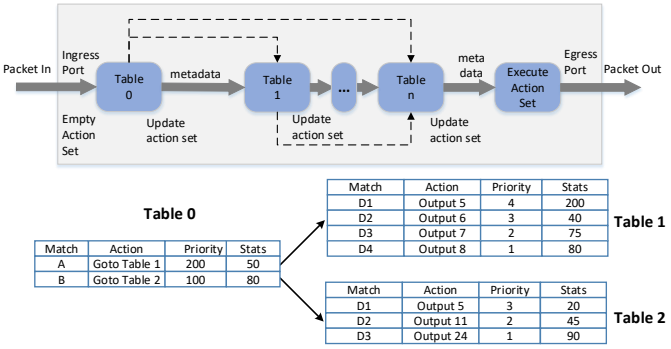


Fig. 1. Forwarding table structure

Index	Match Fields	Action	Priority	Statistics
1	MAC = A, IP = B	Port 1	200	40
2	MAC = C	Port 4	90	80
3	IP = B	Port 12	70	60

SCF_f	Lower bound	Upper bound
MAC = A, IP = B	40	40
IP = B	100	180
MAC = D, IP = B	0	60

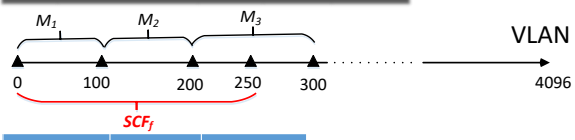
Fig. 2. A single forwarding table ODCI estimation example

detailed multi-table structure with three linked tables. These tables collectively form a forwarding set to match against all incoming packets and execute associated actions. Since these chained tables behave as a packet processing pipeline, they are also called *forwarding pipeline*. In this forwarding pipeline, the actions in "Table 0" contains "Goto Table" attribute which means the matched packets of this forwarding entry will be passed to another table for further processing. A packet with attributes "A" and "B" will be sent to Table 1 and Table 2 for further processing, respectively. Note that in each table, every single entry is associated with a statistic, which means a packet's traffic will be counted more than one time if it travels along multiple tables.

Our DECISION solution in section IV begins from a single forwarding table and then extends to the multi-table scenario. A single flow table is composed of multiple flow entries which is illustrated as a row in a table. Take the table in the top half of Fig. 2 as an example, there are a total three FTEs in which "Match fields", "Action" and "Priority" are OpenFlow's intrinsic attributes. While each FTE is always associated with a statistic, the "Statistics" is not part of a FTE. For the sake of easy reference to a specific FTE, an "Index" column is also added into this table.

Using the forwarding table example in Fig. 2 which contains three FTEs indexed 1-3 (the top half) as shown, the ODCI for the flowset with SCF as $\{MAC = A, IP = B\}$, $\{IP = B\}$ and $\{MAC = D, IP = B\}$ are $[40, 40]$, $[100, 180]$ and $[0, 60]$, respectively (bottom half in Fig. 2). Take the second SCF $\{IP = B\}$ as an example, the packets with IP address as B must match FTE 1 and 3, thus the LODCI is the sum of FTE 1's and FTE 3's statistics ($40 + 60 = 100$). However, the packets with MAC address as C and IP address as B will

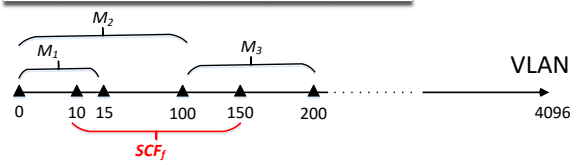
Index	Match Fields	Action	Priority	Statistics
1	VLAN = [1 – 100]	Port 1	200	40
2	VLAN = [101 – 200]	Port 4	90	80
3	VLAN = [201 – 300]	Port 12	70	60



SCF_f	Low bound	Upper bound
VLAN = [1-250]	40+80	40+80+60

Fig. 3. A traffic estimation example on independent FTE set

Index	Match Fields	Action	Priority	Statistics
1	VLAN = [1 – 15]	Port 1	200	40
2	VLAN = [1 – 100]	Port 4	90	80
3	VLAN = [101 – 200]	Port 12	70	60



SCF_f	Low bound	Upper bound
VLAN = [11-150]	0 ?	80+60 ?

Fig. 4. A traffic estimation example on non-independent FTE set

only match FTE 2, which means the UODCI is the sum of all the three FTE's statistics in this table $(40 + 80 + 60 = 180)$.

IV. DECISION

A. Single-Table Traffic Statistics Estimation

Intuitively, LODCI of a given flowset f is the sum of the statistics for all FTEs whose match-field set is a subset of f's SCF (denoted as SCF_f). The UODCI, however, is a set cover problem i.e. to find a collection of FTEs whose union match-field set is a superset of SCF_f. This approach is referred to as Simple-Subset-Superset (S^3) in the reminder of this paper. Take the forwarding table in Fig. 3 as an example, the match fields of the first two FTEs M_1 and M_2 are subsets of flowset f , thus the lower bound of flowset f is the sum of these two FTEs' statistics (i.e. 120). The union of all the three sets $\{M_1, M_2, M_3\}$ is a superset of SCF_f, thus the upper bound of flowset f is the sum of all the statistics (i.e. 180).

In the example illustrated in Fig. 3, all match fields are independent, (i.e. their match fields have no intersections). However, for FTEs with intersections, the S^3 approach cannot guarantee that an optimal lower and upper bound can be found. Figure 4 demonstrates a forwarding table with two FTEs (Index 1 and 2) whose match fields intersect. For a flowset $SCF_f = \{$ VLAN ranges from 11 to 150$\}$, there no FTE whose match-field set is the subset of f and the estimate is always 0. For the upper bound, the minimum covering set for f are FTEs with Index 2 and 3's match-field sets, thus the upper bound is the sum of these two FTEs' statistics (i.e. $80 + 60 = 140$).

Unfortunately, the ODCI estimation based on the S^3 approach for the case in Fig. 4 is incorrect. As mentioned earlier, all incoming packets to a switch match against the FTEs in a single table in the sequence of decreasing priority. Once a packet matches an FTE, the corresponding action will be performed and the remaining FTEs will be ignored. As an example, a packet with VLAN between 1 and 15 will never match against FTE 2 in Fig. 4, thus the effective VLAN id of FTE 2 ranges from 16 to 100 rather than from 1 to 100. Hence all FTE 2's matching packets belongs to the estimated flowset whose $SCF_f = \{VLAN = [11, 150]\}$ and the lower bound of flowset f should be 80 rather than 0. Similarly, the union set of the FTE 2 and 3's match fields do not include VLAN id between 11 and 15(inclusive) and it is not the cover set of the given flowset's SCF. The correct LODCI and UODCI of this given flowset should be [80, 180] rather than [0, 140].

Because the proposed S^3 approach is only applicable to the independent FTE sets, the intersections between two FTEs' match fields must be eliminated. The elimination will lead to a large match set for each FTE and thus it is unlikely to be adopted in the real-world traffic estimation. However, it is a rigorous approach and the ODCI will be guaranteed. Besides this, we also propose a pragmatic approach which does not require removing dependencies on all FTEs. It is more practical because it offers a coarse traffic estimate involving far less computation.

1) AS^3 Approach: In a single OpenFlow table, an incoming packet will hit only one FTE no matter how many FTEs it matches. Therefore we have what we call an **actual** match-field (AMF) set which excludes all the match fields of higher-priority FTEs. Once a FTE's original match-field (OMF) set is replaced by its AMF set, these FTEs are independent (within a table) and their positions in a table are exchangeable. This process is called "deprioritisation" because the "Priority" attribute is safely removed without changing forwarding behaviour. In the following discussion, the S^3 approach will be called as AS^3 when applied on AMF and OS^3 when applied on the original match fields.

Fig. 5 illustrates the deprioritisation process with a five-entries example. In Table 1 (the original table), each item is associated with a priority which determines the matching sequence of an incoming packet. To eliminate the dependencies among different FTEs, a virtual attribute "Unmatch Fields" is added in each FTE to exclude the match fields in all FTEs with higher priorities. Thus a FTE's AMF can be expressed by the logical *AND* operation between original "Match Fields" and the new added "Unmatch Fields". Here "Unmatch Fields" is the logical negation of the union of all previous FTEs' OMFs. The deprioritisation preprocessing of a single table is depicted in Algorithm 1 where "$\sum$" represents the logical operation "OR" ($\vee$) and "$\cdot$" means the logical operation "AND" ($\wedge$).

In a deprioritised forwarding table, the original match field of a FTE has been converted into AMF and the AMFs in a table are independent. Algorithm 2 outlines the process of computing the LODCI (lv) and UODCI (rv) from a deprioritised table using AS^3. The LODCI is the sum of the statistics($stat$) for all FTEs whose match-field set is a subset of the given flowset SCF(Scf) (Algorithm 2, lines 7-9). For UODCI, all FTEs are enumerated to find the SCF's intersected FTEs, the sum of these FTEs' associated statistics is

Table 1

Index	Match Fields	Action	Priority	Statistics
1	M_1	A_1	200	S_1
2	M_2	A_2	90	S_2
3	M_3	A_3	70	S_3
...	...	...	...	...
N	M_N	A_N	10	S_N

Table 2

Index	Match Fields	Unmatch Fields	Action	Priority	Statistics
1	M_1	NULL	A_1	X	S_1
2	M_2	M_1	A_2	X	S_2
3	M_3	M_1 M_2	A_3	X	S_3
...	...	...	...	X	...
N	M_N	M_1 M_2... M_{N-1}	A_N	X	S_N

Fig. 5. Single table deprioritisation

Algorithm 1 Single Table Preprocessing: Deprioritisation

1: $T \leftarrow$ input: An Array of FTEs with Match Fields
2: $T \leftarrow$ output: An Array of FTEs with AMF
3: **procedure** $Deprioritisation(T)$
4: $T[0].amf \leftarrow T[0].omf$
5: **for** $i = 1,\ i < T.count,\ i{+}{+}$ **do**
6: $T[i].amf \leftarrow T[i].omf \cdot (\neg(\sum_{j=1}^{j=i-1} T[j].omf))$
7: **end for**
8: **end procedure**

UODCI (Algorithm 2, lines 10-15). If the default FTE (packets sent to controller or dropped) is taken into consideration, the union of all FTEs' match fields is a full set of SCF, thus the UODCI is finite and computable.

Algorithm 2 Single Table ODCI Estimation (AS^3)

1: $Scf \leftarrow$ input: SCF of a given flowset
2: $T \leftarrow$ input: An Array of FTE with AMF
3: lv, $rv \leftarrow$ lv: LODCI, rv : UODCI, $[lv, rv]$ is the ODCI
4: **procedure** $Single\_table\_ODCI(Scf, T, lv, rv)$
5: $lv \leftarrow 0$; $rv \leftarrow 0$; $S[0] \leftarrow Scf$
6: **for** $i = 1,\ i <= n,\ i{+}{+}$ **do**
7: **if** $T[i].amf \subseteq Scf$ **then**
8: $lv \leftarrow lv + T[i].stat$
9: **end if**
10: **if** $S[i] \neq \emptyset$ **then**
11: $S[i] \leftarrow S[i-1] \cap (\neg T[i].amf)$
12: **if** $(T[i].amf \cap S[i-1]) \neq \emptyset$ **then**
13: $rv \leftarrow rv + T[i].stat$
14: **end if**
15: **end if**
16: **end for**
17: **return** lv, rv
18: **end procedure**

2) OS^3 Approach: The operations on SCF and FTEs' match fields are boolean logical operation upon their respective boolean functions. The complexity of a boolean function depends on two factors: width and length. The width means number of literals in a formula-based boolean function or the bits of a value in a truth-table based boolean function while the length represents the number of products in a disjunctive normal form (DNF) or values in a truth table [5].

In deprioritisation, a FTE's AMF is composed of this FTE's original match fields and the compliment of the AMF of the FTEs with higher priority. The length of a AMF is usually far greater than the original match-field set. This introduces significant computation effort when performing the boolean operations on AMF. A more practical approach with DECISION builds ODCI without deprioritisation.

In OS^3, a given flowset f's statistics must not be less than the sum of all FTE's statistics if these FTE's match-field sets are subsets of f's SCF. It means a flowset's lower bound is still able to be estimated based on the subset relation between FTE's match fields and a flowset's SCF. However, for the case that a FTE's AMF rather than its OMF is a subset of a given flowset, this FTE will be excluded from the computation of lower bound. Referring back to Fig. 4, the second FTE cannot be included in the computation of lower bound because its original VLAN range is not a subset of the given SCF_f while its AMF is subsumed into SCF_f. Thus, the lower bound achieved by OS^3 is lower than AS^3, which means that OS^3 cannot provide an optimal DCI in some circumstances. Although AS^3 approach achieves better (greater) lower bound than OS^3, both achieve the same upper bound estimates, and this is given as a proof below:

Proof: The proof of the equivalence of UODCI between AMF and OMF relies on two conditions which must be met in each iteration : i) the value of $S[i]$ is identical for both AMF and OMF in step i; ii) the conditional statement of $(T[i].amf \cap S[i-1]) \neq \emptyset$ (line 12 in Algorithm 2) yields identical result (i.e. *true* or *false*) if $T[i].amf$ replaces $T[i].omf$.

By definition, we have $T[0].amf = T[0].omf$. By induction on k, we have the same $S[k]$: $S_{amf}[k] = S_{omf}[k]$. Let $i = k+1$, for AMF, we have:

$$S_{amf}[k+1] = S[k] \cap (\neg T[k+1].amf)) \quad (1)$$

$$T[k+1].amf = T[k+1].omf \cap (\neg T[0].omf) \cap (\neg T[1].omf) \cdots \cap (\neg T[k].omf) \quad (2)$$

where Eq. 1 and Eq. 2 come from Algorithm 2 Line 11 and Algorithm 1 Line 6, respectively. For the case of OMF, we have:

$$S[k] \cap T[i].omf = \emptyset \ \forall i \in [0, k], \quad (3)$$

which holds because in the $(k-1)$-th iteration, we assert that $S[k] = S[k-1] \cap (\neg T[k].omf))$. The right hand side of Eq. 1 is expanded based on Eq. 2 to give:

$$(S[k] \cap T[k+1].omf) \cup (S[k] \cap T[0].omf) \cup (S[k] \cap T[1].omf) \cdots \cup (S[k] \cap T[k].omf), \quad (4)$$

and then simplified to:

$$S[k] \cap T[k+1].omf, \quad (5)$$

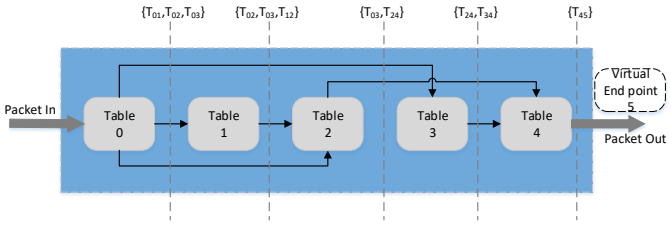


Fig. 6. An example of multi-Table ODCI estimation

which is by definition $S_{omf}[k+1]$. Therefore, substituting Eq. 5 back to Eq. 1 yields to desired result:

$$S_{amf}[k+1] = S_{omf}[k+1]. \quad (6)$$

Similarly, we can also prove the second condition: $T[i].amf \cap S[i-1].amf = T[i].omf \cap S[i-1].omf$ using Eq. 2-5. Thus, Algorithm 2 produces identical UODCI with AMF and OMF. ■

B. Multi-Table Traffic Statistics Estimation

All the FTEs in a switch are constructed either in the form of single table or multiple tables. For a multi-table switch, all packets must match against "Table 0" for subsequent forwarding to other tables via the "Goto Table" action. As illustrated in Fig. 6, a packet has potentially multiple paths to reach its egress port(s). Hence the estimation of a flowset must consider all the potential paths to compute the ODCI. In this example, an arrow line is added between two tables if there exists one FTE in the former table which contains "Goto Table" action to the next table. For a point in a forwarding pipeline, potential paths for a packet passing through this point can be identified by the corresponding table indexes. For example, for any point between "Table 0" and "Table 1", the potential paths for a packet are $\{T_{01}, T_{02}, T_{03},\}$ in which T_{ij} represents a path between "Table i" and "Table j". To take the packets going straightforward from one table to the egress port(s) into consideration, a virtual end point "5" is added. Thus, T_{45} represents the path between "Table 4" and the egress ports, the traffic passing through it includes all packets egressing from physical ports.

At any given point in a forwarding pipeline, the local ODCI is an accumulation of the statistics for all the possible forwarding paths at the same point. Then a global ODCI can be easily selected from these local ODCIs. The process is demonstrated in Algorithm 3 in which $S_{ij} = [lv_{ij}, rv_{ij}]$ represents the local ODCI between Table i and j. In Algorithm 3, we assume that the ODCI for each table has been computed. Thus, a local estimation of a point between two tables is achieved by the accumulating statistics of all the potential paths at that point (Algorithm 3 Lines 6-12). Finally a global optimal estimation is accomplished by the selection of greatest LODCI and least UODCI (Algorithm 3 Lines 13-18).

Algorithm 3 Multiple Table ODCI Traffic Estimation

1: $S_{ij} = [lv_{ij}, rv_{ij}] \leftarrow$ input: local ODCI
2: $N \leftarrow$ input: index of the last table $(0 \le i \le N)$
3: lg, $rg \leftarrow lg$: global LODCI, rg: global UODCI
4: **procedure** $Multiple\_table\_ODCI(S, N, lg, rg)$
5: **for** $k = 0,\ k \le N,\ k{+}{+}$ **do**
6: $tmp\_lg \leftarrow 0$, $tmp\_rg \leftarrow 0$
7: **for** $i = 0,\ i \le k,\ i{+}{+}$ **do**
8: **for** $j = i+1,\ j \le N+1,\ j{+}{+}$ **do**
9: $tmp\_lg \leftarrow tmp\_lg + lv_{kj}$
10: $tmp\_rg \leftarrow tmp\_rg + rv_{kj}$
11: **end for**
12: **end for**
13: **if** $tmp\_lg > lg$ **then**
14: $lg \leftarrow tmp\_lg$
15: **end if**
16: **if** $tmp\_rg < rg$ **then**
17: $rg \leftarrow tmp\_rg$
18: **end if**
19: **end for**
20: **return** lg, rg
21: **end procedure**

V. Evaluation

A. Test bed & Traffic

An evaluation of the DECISION solution was performed on an Open vSwitch (version 2.6.1) which supports 255 flow tables [6]. As illustrated in Fig. 7, three virtual machines (VMs) with a Ubuntu server are created, one is installed with an Open vSwitch as OpenFlow switch (vm2) and the rest two VMs as traffic generator(vm1) and receiver(vm3). In the host OS, Ryu (version 4.11) [7] is installed as a controller to provision OpenFlow rules and retrieve the traffic statistics while Scapy [8] is the traffic generator.

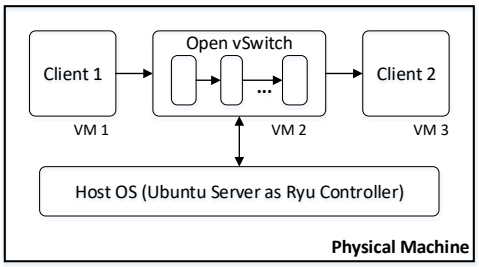


Fig. 7. Test bed environment

In our evaluation, the estimated statistic for a given flowset is compared with the ground truth, the actual traffic, which is measured by installing a new FTE with the exactly same matching fields. This is achieved by reserving tables in the head of a forwarding pipeline for traffic measurement only [2]. In the following description, the new installed FTE and reserved table are called "traffic measurement FTE" (TM-FTE) and "traffic measurement table" (TM-Table), respectively. Figure 8 depicts the design where the tables in the beginning are reserved for measurement and the rests behave as normal packet forwarding.

B. Functional testing

In the DECISION functional testing, we evaluate both correctness and accuracy. A correct traffic estimation means the actual traffic value (av) falls into the range of the estimated

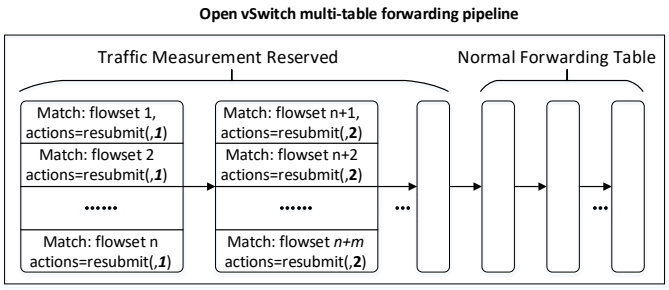


Fig. 8. Reserving Tables for Traffic Measurement

ODCI: $lo \leq av \leq ro$ where lo and ro stand for LODCI and UODCI, respectively.

The accuracy of ODCI reflects how close an estimation is to the actual traffic. In our approach, accuracy is determined by the distance between LODCI and ground truth (the actual traffic, denoted as av in Eq. 7) as well as UODCI and ground truth. In each estimation, the sum of aforementioned distances must be always less than the total traffic passing through a switch, herein a relative distance is chosen to measure the accuracy among different estimations. As expressed in Eq. 7, the accuracy is determined by the relative length of ODCI.

$$\begin{aligned} accuracy &= \left\{1 - \frac{|av - lo| + |ro - av|}{total}\right\} \times 100\% \\ &= \left\{1 - \frac{|ro - lo|}{total}\right\} \times 100\%. \end{aligned} \tag{7}$$

The functional test results in different scenarios are presented in Table I. In a single-table scenario, a FTE consists of more than one match field, for example, a combination of destination MAC (DMAC) and destination IP address (DIP) (the second and third row in Table I). However, in a multi-table scenario, one table for one match field [9]. For example, the first table matches the incoming packets' MAC address and the second table matches IP address, etc. Therefore a single match field is shown in the multi-table scenario (rows 4-7 in Table I). The columns "FTE", "Traffic Pattern", and "FTE Count" denote the FTE's match fields, the pattern of the traffic generated by Scapy, and the number of FTEs in the table, respectively. The fifth column, "Total Traffic", is the recorded traffic passing through the table for each test, i.e., the value of $total$ in Eq. 7. The "Ground Truth" is the actual traffic (av in Eq. 7) which is measured by TM-FTE (Fig. 8). Each set of tests are repeated ten times and the average accuracy is computed with 95% confidence intervals falling between 0.2% – 0.7% (for all results in Table I).

From the observation of all the values in the column of "Ground Truth", the ground truth always falls between the range [LODCI, UODCI], which means the traffic estimation results are correct. For the same FTE type, a total of 2^{14} and 2^{15} FTEs (denoted by column "FTE Count") are installed into Open vSwitch respectively. The accuracy increases with increasing number of FTEs because more FTEs are likely to produce a simple-subset-superset that matches a given flowset's SCF.

Another observation is that the accuracy decreases when the match fields base increases (the number of FTEs held constant). Here the base is the total number of possible combinations for a given type of match field. For example, the base of the "DIP+DMAC" in the single table scenario is 2^{16+24} while the base of "DIP" in the multi-table scenario is only 2^{16}, thus the estimation accuracy with a total of 2^{15} FTEs on "DIP" (79.8%) will be better than "DIP + DMAC" (11.1%) because the probability of finding a closer superset and subset in the former case is higher.

C. Discussion

1) Accuracy: In the functional test results illustrated as Table I, the accuracies vary from 79.8% to 6.5%. The low accuracies in our experiments are the worst case scenarios and due to the randomly generated match fields of estimated flowsets. In typical traffic estimation uses, such scenarios are unlikely to occur because the flowset to be estimated is usually highly correlated with existing FTEs and are more likely to have narrower UODCI and LODCI than the randomly selected flowset.

The estimation accuracies increase when increasing number of FTEs, and this is easily observed in Fig. 9 with the experiment on a DIP match field whose first 16 bits are fixed. In other words, its base size is 2^{16}. The ratio of the number of FTE to the base explains why the accuracy of the first several tests (number of FTEs ranges from 2^1 to 2^6) are very low ($\leq 10\%$). A low ratio means the low probability to find the appropriate subsets and supersets for a given flowset. This experiment clearly indicates the high positive correlation between the number of FTEs and accuracy.

2) Estimation Time: In this set of experiments, the time to find ODCI for a given flowset on various match fields is evaluated. The time to complete a round of traffic estimation can be expressed as:

$$T_{est} = T_{comm} + T_{switch} + T_{controller} \ ,$$

where T_{comm} represents the communication time between controller and switches, T_{switch} denotes the querying time for traffic statistics and $T_{controller}$ denotes the time to process traffic statistics at the controller side. Recent measurement studies have shown that both T_{comm} and T_{switch} are in order of milliseconds [10], [11] compared to seconds for $T_{controller}$, and can be safely ignored in computing measures of efficiency [2]. The processing time at the controller, $T_{controller}$, is proportional to the number of FTEs. The results in this subsection are reported based on tests conducted on a Dell OptiPlex 9020 server with Intel i7 Quad-Core CPU and 8GB RAM.

Take the far right three bars in Fig. 10 as an example, the estimation time for a given flowset in a 2^{20} long FTE table costs around 900ms. Here 2^{20} is chosen because it is in the same order as the maximum FTE entries (1×10^6) in an Open vSwitch table. Thus it can be concluded that a round of traffic estimation is able to be completed in one second on an entry-level server.

TABLE I
FUNCTIONAL TEST RESULT

Scenario	FTE	Generated Traffic Pattern	FTE Count	Total Traffic (Bytes)	Ground Truth (Bytes)	LODCI (Bytes)	UODCI (Bytes)	Accuracy
Single Table	DMAC, DIP	IP(dst=RandIP("10.1.*.*"))/ Ether(dst=RandMAC("11:22:33:*:*:*"))	2^{14}	$6.55 * 10^6$	$1.28 * 10^2$	$6.4 * 10^1$	$6.03 * 10^6$	6.5%
	DMAC, DIP	IP(dst=RandIP("10.1.*.*"))/ Ether(dst=RandMAC("11:22:33:*:*:*"))	2^{15}	$6.55 * 10^6$	$1.92 * 10^2$	$6.4 * 10^1$	$5.82 * 10^6$	11.1%
Multiple Table	DMAC	Ether(dst=RandMAC("11:22:33:*:*:*"))	2^{14}	$6.55 * 10^6$	$1.92 * 10^3$	$1.92 * 10^2$	$5.27 * 10^6$	19.5%
	DMAC	Ether(dst=RandMAC("11:22:33:*:*:*"))	2^{15}	$6.55 * 10^6$	$2.56 * 10^3$	$2.56 * 10^2$	$4.96 * 10^6$	24.2%
	DIP	IP(dst=RandIP("10.1.*.*"))	2^{14}	$6.55 * 10^6$	$3.84 * 10^3$	$3.2 * 10^2$	$5.12 * 10^6$	51.4%
	DIP	IP(dst=RandIP("10.1.*.*"))	2^{15}	$6.55 * 10^6$	$4.96 * 10^3$	$7.04 * 10^2$	$1.32 * 10^6$	79.8%

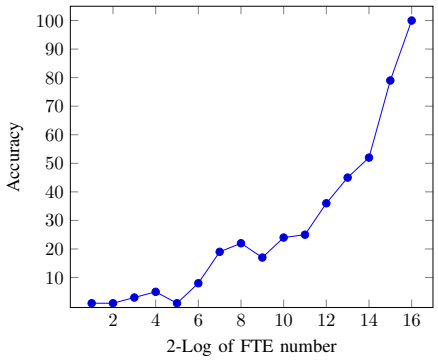


Fig. 9. Increasing accuracy with number of FTE

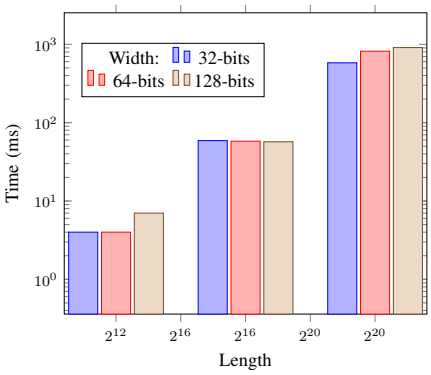


Fig. 10. Time to complete an estimation

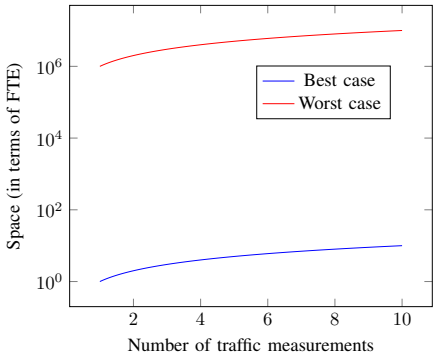


Fig. 11. TM-FTE storage space in traditional measurement

3) FTE Storage Space: Compared to the traditional approaches which relies on the TM-FTE to monitor the traffic of a flowset, the DECISION solution achieves the similar goal by evaluating existing entries of a flow table. As illustrated in Fig. 8, TM-Tables are reserved for traditional measurements. This can be achieved by installing all TM-FTEs in "Table 0". However, for multiple parallel measurements, any intersecting TM-FTEs must be installed in a different table. Otherwise, a measurement by a TM-FTE with lower priority is incorrect because incoming traffic might hit higher-priority TM-FTEs and traverse other tables.

The FTE storage space (measured in number of FTEs) required by the TM-FTE is illustrated in Fig. 11, from which we can observe that the storage space increase up to six orders of magnitude when the measurements of intersecting flowsets occur. In the best case, all the match fields of estimated flowsets are independent and these TM-FTEs are installed in a single table; in the worst case, the match fields of every two estimated flowsets intersect and each TM-FTE occupies a new table.

VI. RELATED WORK

ProgME (Programmable Network MEasurement) [1] presented a framework to measure a flowset defined according to application requirement. A statistics query is processed by a query answering engine which maps a flowset to unique flows whose statistics can be retrieved directly from hardware (TCAM). Even though the scenario of this paper and our research vary, they share the same principle: to measure an arbitrary set of flows' traffic based on the ready statistics. However, the query answering engine in ProgME is not suitable to SDN due to the fact that the forwarding pipeline in SDN is more flexible and complicated than in traditional networking. Besides, FTEs are constructed in a complex data structure in the form of either single-table or multiple-table. Thus in SDN, the simple mapping between a flowset and individual flows proposed by ProgME is not applicable any more.

Most existing works on networking traffic estimation in SDN rely on the manipulations of the configuration and forwarding policies (rules) in switch [12]–[16]. All the surveyed works perform traffic measurement and estimation by installing application specified rules and then retrieving their statistics, though their capabilities and scenario vary significantly. As depicted in Table II, similar to ProgME [1], our proposed DECISION offers the queries for an arbitrary flow set without relying on the manipulation of FTEs. Thus the dependencies on hardware and the conflicts between original packet forwarding FTEs and the monitoring FTEs have been eliminated.

VII. CONCLUSION

Software defined networking enables a controller to have more comprehensive understanding on the status of a network by retrieving real-time flows' statistics. Besides an individual flow's statistics, a controller is also interested in the statistic of a flowset. The existing literatures address this requirement by installing and monitoring new customized FTE(s) . However, this process is not always required because an estimation for the traffic of a flowset can be achieved by mining on the

TABLE II
LITERATURE REVIEW: NETWORKING TRAFFIC MEASUREMENT/ESTIMATION

Project	Flow Type	Pipeline Manipulation	Hardware Dependencies	Scenario
ProgME [1]	Flowset	Not Required	N/A	Non-SDN, Heavy Hitter Detection
iSTAMP [12]	Aggregated	FTE	TCAM	SDN, Generic Purpose
DCM [13]	Aggregated	FTE	TCAM/FPGA	SDN , Generic Purpose
OpenSketch [14] [15]	Flow	Sketch	TCAM/FPGA	SDN , Generic Purpose
OpenWatch [16]	Flow	FTE	Not Mentioned	SDN, Anomaly Detection
DECISION	Flowset	Not Required	N/A	SDN, Generic Purpose

already existing FTEs and their statistics. Based on the analysis of the relation between a flowset's specified common fields and the existing FTEs' match fields, this paper proposes a deterministic confidence interval estimation approach for an arbitrary flowset. The evaluation and analysis based on the results of our Open vSwitch based testbed have proved the feasibility and efficiency of our proposed approach. The future work includes the deployment and evaluation on a realistic large-scale data centre, which is now under negotiation with a local cloud service provider.

VIII. ACKNOWLEDGEMENT

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

REFERENCES

[1] L. Yuan, C.-N. Chuah, and P. Mohapatra, "Progme: Towards programmable network measurement," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 115–128, Feb. 2011.

[2] L. Yang, B. Ng, and W. K. G. Seah, "Heavy hitter detection and identification in software defined networking," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, Aug 2016, pp. 1–10.

[3] V. J. Easton and J. H. McColl, "Statistics glossary v1. 1," 1997.

[4] "Openflow switch specification, version 1.5. 1," *Open Networking Foundation*, December 19, 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf

[12] M. Malboubi, L. Wang, C. N. Chuah, and P. Sharma, "Intelligent sdn based traffic (de)aggregation and measurement paradigm (istamp)," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 934–942.

[5] F. M. Brown, *Boolean reasoning: the logic of Boolean equations*. Springer Science & Business Media, 2012. [Online]. Available: http://www2.fiit.stuba.sk/~kvasnicka/Free%20books/Brown_Boolean%20Reasoning.pdf

[6] The Linux Foundation, "Open vswitch, an open virtual switch," Available: http://docs.openvswitch.org/en/latest/contents/, accessed 2017-03-20.

[7] NTT, "Ryu sdn framework," Available: https://osrg.github.io/ryu/, accessed 2017-03-20.

[8] P. L. Philippe Biondi, Guillaume Valadon, "Scapy," Available: http://www.secdev.org/projects/scapy/, accessed 2017-03-20.

[9] ONF, "The benefits of multiple flow tables and TTPs, version number 1.0," February 02, 2015. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_Multiple_Flow_Tables_and_TTPs.pdf

[10] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc of the USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, San Jose, CA, 24 Apr 2012, 6 pages.

[11] R. Edwards, Y. Ye, A. Kaul, and J. Yu, "Characterization of SDN switch response time in proactive mode," in *Proc of the Open Networking Summit (ONS)*, Santa Clara, CA, USA, 2-4 March 2014, poster.

[13] Y. Yu, C. Qian, and X. Li, "Distributed and collaborative traffic monitoring in software defined networks," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 85–90.

[14] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 29–42.

[15] L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches." in *Hot-ICE*, 2011.

[16] Y. Zhang, "An adaptive flow counting method for anomaly detection in sdn," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '13. New York, NY, USA: ACM, 2013, pp. 25–30.