sensors

MDPI

Article

Entry Aggregation and Early Match Using Hidden Markov Model of Flow Table in SDN

Cheng Wang [1] and Hee Yong Youn [2,*]

[1] Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, Korea; lighting91@skku.edu
[2] College of Software, Sungkyunkwan University, Suwon 16419, Korea
* Correspondence: youn7147@skku.edu

Received: 3 April 2019; Accepted: 17 May 2019; Published: 21 May 2019

Abstract: The usage of multiple flow tables (MFT) has significantly extended the flexibility and applicability of software-defined networking (SDN). However, the size of MFT is usually limited due to the use of expensive ternary content addressable memory (TCAM). Moreover, the pipeline mechanism of MFT causes long flow processing time. In this paper a novel approach called Agg-ExTable is proposed to efficiently manage the MFT. Here the flow entries in MFT are periodically aggregated by applying pruning and the Quine–Mccluskey algorithm. Utilizing the memory space saved by the aggregation, a front-end ExTable is constructed, keeping popular flow entries for early match. Popular entries are decided by the Hidden Markov model based on the match frequency and match probability. Computer simulation reveals that the proposed scheme is able to save about 45% of space of MFT, and efficiently decrease the flow processing time compared to the existing schemes.

Keywords: SDN; entry aggregation; Quine–McCluskey Algorithm; match frequency and probability; Hidden Markov Model

1. Introduction

In software-defined networking (SDN) [1,2] efficient network management is achieved by separating the control plane from the data plane, and providing programmable interfaces to the application layer to flexibly enable the customization of the network management functions. The traditional legacy network services often require various processing of data packets such as routing, monitoring, access control, and load balancing. If all the services are provided in a single network infrastructure, the development of the network will be very complicated and adapting to a variety of functional requirements will be difficult. In this regard, SDN provides a new direction of networking services and drastically changes the traditional network with distinctive features including the separation of controller and forwarding unit, virtualization, programmable service, dynamic reconfiguration, and centralized management [3].

SDN is an innovative movement in the field of computer networks. It stems from a series of projects on networks such as active networks, software switches, 4D (Decision, Dissemination, Discovery, and Data) networks, and traditional telephone networks [4,5]. The network infrastructure has been evolving from the approach of vertical integration to horizontal one to be more intelligent. Intelligent control of data forwarding units (switches, routers, etc.) has been unified into the control plane [2]. With SDN the network devices of the data plane only forward the data, while they are directly managed by the control plane. As a result, the control plane can get the global information of the network such as connection topology, link state, etc., and thus proper decisions on forwarding the flows can be made. The network is programmable through a unified interface provided to control the behavior of the whole network [6]. The programmable interface greatly improves the flexibility of network management and data forwarding, and enables the network to dynamically change the path

of the flow. In SDN, the physical characteristics of the networking devices are concealed so that the control plane can provide a unified management and services for high level applications [7]. At the same time, the data forwarding devices are abstracted from the control layer to ensure the mobility, and reduce future investment of the devices.

Among the main issues with SDN such as switch designs [8,9], distributed controller platforms [10,11], resilient communication [12,13], and security [14,15], flow table management is one of the primary tasks directly influencing the performance. From OpenFlow 1.3 [16,17], there were several improvements including multiple flow tables (MFT). In SDN, the OpenFlow switches just forward the flows through the MFT which are composed of data packets, and the operation is managed by OpenFlow controllers. Even though MFT plays an important role in SDN, two challenging issues exist which are memory overhead and flow forwarding delay. The memory overhead is high due to the need of ternary content addressable memory (TCAM) and fast forwarding [18]. If the number of entries in the table is limited too much for reducing the overhead, the flow match rate will be very low. Meanwhile, the pipeline mechanism employed with MFT causes forwarding delay as each flow has to go through the flow table one after another, and the action corresponding to the matched entry is executed at the end of the pipeline. There exist various studies dealing with such issues of MFT. For example, the Flow Table Reduction Scheme [18] (FTRS) aggregates the flow entries of the same action and destination addresses. In [19], the Multi-Stage OF (MsOF) switch model is proposed to save memory space and reduce forwarding time by deploying the tables each requiring less memory space. However, the FTRS works well only in local area network (LAN) environment and the implementation of MsOF is relatively complicated.

In this paper we advocate a novel approach called Agg-ExTable which efficiently reduces the memory overhead and flow forwarding time of MFT. Here pruning and the Quine–McCluskey (QM) algorithm are periodically applied to aggregate the flow entries in each flow table, and a table called ExTable is put in front of the pipeline which contains the flow entries of high expected match probability. Its flow entry is dynamically determined using a Hidden Markov model constructed based on the number of transitions and match probability of the entries. The QM algorithm allows an efficient usage of TCAM by reducing the size and power consumption, while ExTable allows quick match and execution of the incoming flows. Computer simulation reveals that the proposed scheme substantially reduces the memory size and flow processing time compared to the TCAM-based size reduction scheme [20] and the scheme merging and migrating the flow entries based on directed acyclic graph (DAG) [21]. The improvement gets more significant as the flow arrival rate rises. The main contributions of the paper are summarized as follows:

- The flow entry aggregation problem is simplified by transforming it to a logic minimization problem, which is effectively solved by the QM algorithm.
- The ExTable scheme substantially reduces the flow processing time by placing a table containing the entries of high match probability up front.
- The match frequency and match probability of the entries are handled with the Hidden Markov model to decide the likelihood of the match.

The remainder of the paper is organized as follows. Section 2 provides an overview of TCAM used for the implementation of MFT, Hidden Markov model, and MFT-based switching. In Section 3 the proposed Agg-ExTable approach is presented, along with the analytical model of the flow processing time based on queueing theory. Section 4 is for the performance evaluation of the proposed schemes. Finally, Section 5 concludes the paper and outlines the future research direction.

2. Related Work

2.1. TCAM

TCAM [20,22] is usually employed to speed up the table look up operation. However, the size is always limited due to high cost. TCAM allows a third matching state of 'X' or 'don't care' for one or

more bits in the stored entry, allowing the flexibility in the search. For example, a TCAM may have an entry of '101XX' which will match to '10100', '10101', '10110', or '10111'. The additional state is typically implemented by adding a mask bit to the corresponding memory cell. If a bit of the mask is '0', the bit of corresponding entry is 'don't care'. In Table 1, for example, if the first five bits of a flow are '10011' or '10111', it matches E_1.

Table 1. An example of flow table with TCAM.

No.	Entry/Mask								Action
E_1	1	0	X	1	1	X	X	X	Forward to 1
	1	1	0	1	1	0	0	0	
E_2	0	X	0	0	X	X	X	0	Forward to 2
	1	0	1	1	0	0	0	1	
E_3	1	1	X	X	1	1	X	X	Forward to 5
	1	1	0	0	1	1	0	0	
E_4	1	X	X	X	0	X	X	X	Forward to j
	1	0	0	0	1	0	0	0	
E_5	X	X	X	1	1	X	X	X	Forward to k
	0	0	0	1	1	0	0	0	

Another important feature of TCAM is that '0' and '1' in the mask is not required to be continuous. For E_1 in Table 1, the third bit of the entry is 'don't care', and thus the flow whose first three bits are '100' or '101' matches this entry. The proposed aggregation method takes advantage of this feature of TCAM.

In reference [20], two techniques, pruning and ESPRESSO-II based mask extension, are proposed to compact traditional routing table stored in TCAM. They allow a smaller TCAM reducing the size and power consumption.

2.2. Quine–McCluskey (QM) Algorithm

The Quine–McCluskey (QM) algorithm [23–25] was developed for simplifying logical functions. The QM algorithm is effective for implementation since it has tabular form, and it also provides a deterministic method checking if the logical function is minimal. There exist mainly four definitions in using the QM algorithm:

- Minterm: an expression in which all variables of the logical function appears once.
- Implicant: an aggregation of minterms in the logical function.
- Prime implicant: an implicant that cannot be covered by a more simplified implicant.
- Essential prime implicant: prime implicants that cover an output of the logical function for which no combination of other prime implicants is able to cover.

The procedure of the QM algorithm consists of two steps:

- Step 1: Find prime implicants from the minterm table.
- Step 2: Find the essential prime implicants and other necessary prime implicants to cover the logical function.

In the proposed scheme the QM algorithm is employed for mask extension which is transformed to be a logic minimization problem.

2.3. Hidden Markov Model (HMM)

HMM [26–29] is used in various fields such as language recognition, reinforcement learning, and bioinformatics. An HMM is a finite discrete time Markov model in which the system is assumed

to be a Markov process with hidden state. It can be defined as a triple $\lambda = (\pi, A, B)$, where π is the initial probability distribution, A is transition probability matrix, and B is a sequence of observation likelihoods (emission probabilities). Specifically, an HMM is defined by the following components [30]:

- $Q = \{q_1, q_2, \ldots, q_N\}$ is a set of states where N is the number of hidden states, and q_t is the hidden state at time t.
- $O = \{o_1, o_2, \ldots, o_T\}$ is a set of observations where T is the number of observations. o_t is the observable state at time t. Each observation is drawn from a vocabulary $V = \{v_1, v_2, \ldots, v_M\}$, where M is the number of observation values.
- $A = \{a_{11}, a_{12}, \ldots, a_{N1}, \ldots, a_{NN}\}$ is an $N \times N$ transition probability matrix. a_{ij} $(1 \le i, j \le N)$ represents the probability of changing from state_i to state_j. Here,

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i), \tag{1}$$

while,

$$a_{ij} \ge 0, \ \forall i, j, \tag{2}$$

$$\sum_{j=1}^{N} a_{ij} = 1, \ \forall i. \tag{3}$$

- $B = \{b_j(k)\}(1 \le j \le N, 1 \le k \le M)$ is an $N \times M$ emission probability matrix. $b_j(k)$ represents the probability of an observation o_t being generated from state_j. Here,

$$b_j(k) = P\left(o_t = v_k | q_t = s_j\right), \tag{4}$$

while,

$$\sum_{k=1}^{M} b_j(k) = 1, \ \forall j. \tag{5}$$

- $\pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$ is the initial probability distribution over hidden states. π_i is the probability that the Markov chain will start in state_i. Here,

$$\pi_i = P(q_1 = s_i), \tag{6}$$

while,

$$\sum_{i=1}^{N} \pi_i = 1, \ \forall i. \tag{7}$$

A generic HMM is illustrated in Figure 1, where $H_i (i = 1, 2, 3, \ldots, T)$ is the set of hidden states and $O_i (i = 1, 2, 3, \ldots, T)$ is the set of observations. The Markov process, located above the observable states, is determined by the current state and A. Only O_i is able to be observed, which is determined by the hidden states of the Markov process and B.

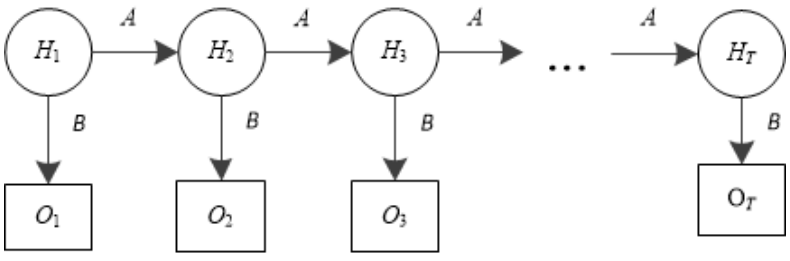

Figure 1. The state diagram of the Hidden Markov model.

2.4. MFT

The pipeline processing of OpenFlow [16,17] is depicted in Figure 2. An OpenFlow switch is required to have one or more flow tables, while a single flow table is used only for relatively simple network.

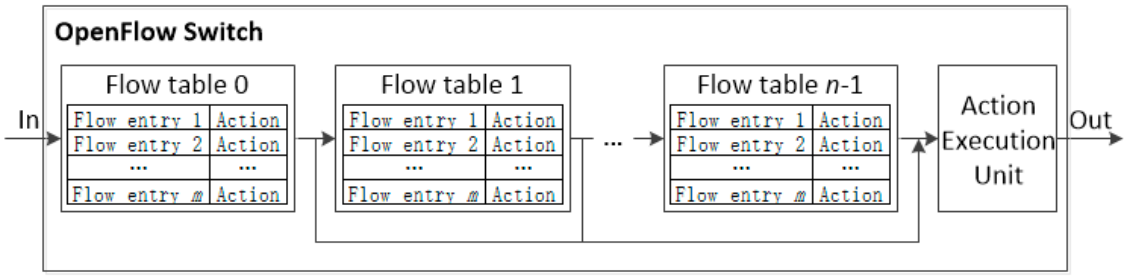

Figure 2. The pipeline processing with MFT.

The flow tables of an OpenFlow switch are sequentially numbered, starting from 0. The pipeline processing always starts from the first flow table where the packet is matched against its flow entries. Other flow tables may be used depending on the outcome of the match. The pipeline processing occurs only in the forward direction. If a flow entry matches, the instruction set included in that entry is executed at the Action Execution Unit (AEU) located at the end of the pipeline.

The existing flow entry aggregation techniques can be classified into two types according to the location of aggregation, in packet classifier or in OpenFlow switch flow table. The TCAM Razor [31] is a systematic flow aggregation algorithm which uses the decision diagram to minimize the TCAM rules required for packet classification. However, the execution time grows rapidly when the number of flow entries increases. The Fast Flow Table Aggregation [32] (FFTA) and FTRS also support entry aggregation in SDN. FFTA is an offline aggregation scheme based on bit weaving, which applies ORTC (Optimal Routing Table Constructor) after cutting the entries using a binary search tree. The FTRS aggregates the flow entries according to the destination IP address, instead of the match field. It achieves a good compression ratio with different topologies. However, FFTA causes coarse traffic statistics due to the mixture of all the entries, while FRTS has a high probability of flow table overflow.

The schemes of [19,33] aim to implement effective forwarding with MFT. In reference [21], an algorithm called migrating flow rules (MILE) was proposed which merges and migrates the flow entries to reduce the number of flow table lookup operations by employing directed acyclic graph (DAG). The dependencies of the flow entries are handled using DAG, where interdependent entries are grouped and migrated as a whole. Using the 2Q LRU replacement algorithm, the recently accessed entries are replaced at Flow Table_0 to be matched early. With the Multi-Stage OF (MsOF) [19] switch model, more tables each requiring less memory space are deployed. Here a processor is implemented in each flow table, and multiple pipeline operations occur in parallel so that several flows can be matched at the same time. The spatial and temporal complexity were examined using queuing theory. The implementation of MsOF is relatively complicated and needs large networking resources.

In order to improve the efficiency of the MFT, both the TCAM implementation and flow match probability are focused in this paper. The proposed scheme aggregate the entries using the pruning and QM algorithm to minimize the space of TCAM, while the saved memory space is used to store popular entries. With the HMM constructed based on the match frequency and match probability of the entries, the presumably popular flow entries are selected and put in the ExTable located in front of the flow table. The pruning and QM algorithm aim to maximize the efficiency of the TCAM by reducing the size and power consumption. Maintaining high match rate with ExTable substantially decreases the overall flow processing time, via early match and execution of the incoming flows. The proposed scheme is presented in the next section.

3. The Proposed Scheme

3.1. The Structure

The proposed Agg-ExTable scheme allows entry aggregation and fast pipeline operation using ExTable holding popular flows. It works in two phases. In Phase 1, the proposed entry aggregation algorithm is executed periodically to reduce the size of TCAM. Then in Phase 2, the saved memory space is utilized to set up ExTable in front of the pipeline, keeping popular entries. Here the HHM is used to decide the popularity of the flow entry. The structure of the proposed MFT pipeline of an OF switch is illustrated in Figure 3. Observe that there exist two paths; express path for popular flow and regular path for nonpopular flow. When a flow arrives at a switch, it is first parsed for the matching with the ExTable. Upon a match, the actions of the flow entry are sent to AEU. Otherwise, the match operation is continued with the other flow tables.

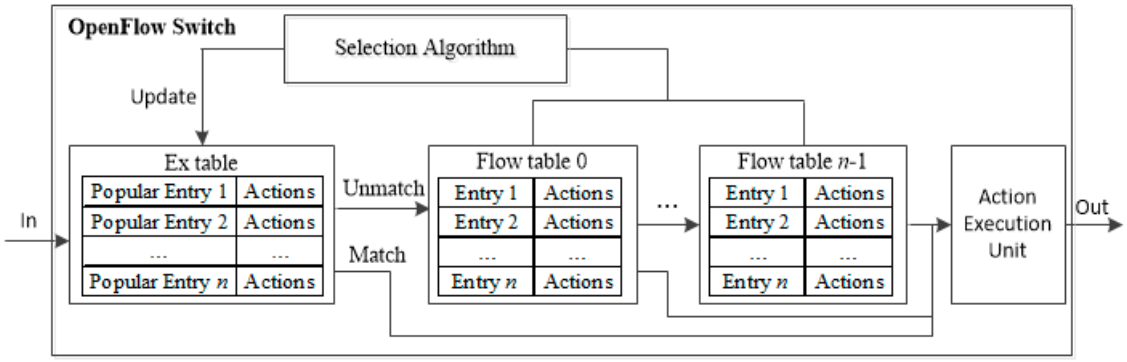

Figure 3. The structure of the proposed scheme.

3.2. Aggregation of Entry

The number of possible flow paths in a flow table is typically small because only a limited number of interface cards can fit into the switch chassis. In contrast, the number of forwarding entries is quite large, in the range of several thousands. Considering this disparity, a scheme reducing the size of flow table is developed which involves two techniques presented below.

3.2.1. Pruning of Redundant Entries

Pruning is a technique eliminating some redundant entries [20]. To facilitate the discussion, some terms are defined as follows. Notice that the match fields of a flow entry may have different lengths.

- Assume that *entry_P* is the parent of *entry_Q*, *Lp* is the length of *entry_P*, and $P(i)$ is the ith bit of *entry_P*. Then the following three conditions hold: (a) $L_P < L_Q$; (b) For all $i(1 < i < L_P)$, $P(i) = Q(i)$; (c) There is no Q' such that $L_P < L_{Q'} < L_Q$; and $Q'(i) = Q(i)$ for all $i(1 < i < L_P)$.
- *Entry_P* is identical to *entry_Q* if same actions are executed for the matched packet.

If P is identical to Q, Q is a redundant flow entry. Assume that Q matches a flow. Then the flow will match P as well by the definition. If Q is removed from the flow table, P becomes the matched entry. As P and Q have the same actions, removing Q makes no difference. Note that the technique is general enough that it can be used with any entry lookup algorithm regardless of the type of the flow table.

3.2.2. QM-Based Mask Extension

The second technique exploits the flexibility offered by the TCAM hardware. TCAM allows arbitrary mask, in other words, the bits of 1s or 0s do not require to be continuous.

Table 2 shows an example of mask extension. E_1 and E_2 both have the same action of 'Forward to 1'. It is possible to combine E_1 and E_2 into one single entry with the prefix of 1100 and mask of 1101. The 0 at bit 3 in the mask allows combining E_1 and E_2 into a same entry. The aggregated version of the original flow table with the mask extension technique is shown in the right-hand side. The table size has been reduced to 3 from 5.

Table 2. An example of mask extension.

Before					After			
No.	**Prefix**	**Mask**	**Action**	**Mask extension**	**No.**	**Prefix**	**Mask**	**Action**
E_1	1100	1111	Forward to 1	⇨	$E_{1\&2}$	1100	1101	Forward to 1
E_2	1110	1111	Forward to 1		E_3	1000	1000	Forward to 2
E_3	1000	1000	Forward to 2		$E_{4\&5}$	1101	1011	Forward to 3
E_4	1101	1111	Forward to 3					
E_5	1001	1111	Forward to 3					

Note that the mask extension is equivalent to the logic minimization problem [20]. The problem is that 'given a set of entries with the same action, find a set of minimal covers.' Such logic minimization problem [33] is a non-deterministic polynomial (NP) complete problem, and there exist mainly three kinds of methods used for its solution.

- Karnaugh mapping [34]: It is simple but when the number of variables is larger than six, it becomes very complex.
- Quine–McCluskey (QM) algorithm [23–25]: It is functionally identical to Karnaugh mapping, but the tabular form makes it more efficient to be used with a computer algorithm, supporting any number of variables. It also provides a deterministic method checking if the logical function is minimal.
- Espresso logic minimizer [35,36]: It can produce a solution fast but cannot guarantee optimal result.

Here the QM algorithm is employed for mask extension. Algorithm 1 shows the proposed entry aggregation scheme with the QM algorithm-based mask extension. Here, $E(l,a)$ is the set of original entries having the same length of l and action of a. $A(l,a)$ is the result of QM algorithm.

Algorithm 1. Entry aggregation with mask extension

```
//n is the number of original entries
//m is the number of entries having the same length and action
1: Begin
2: Input entry[i]
3: for i from 0 to n
4:   if have same entry[i].l and entry[i].a
5:     move to E(l,a)
6: End for
7: for i from 0 to m
8:   A(l,a) = QMminimize(E(l,a))
9: end for
10: remove E(l,a)
11: install A(l,a)
12: end
```

An example of the proposed mask extension scheme is shown below. In Table 3, there are 11 entries with different actions. After selecting the entries having the same action, the entry aggregation problem is simplified into the following minimization function:

$$F(A,B,C,D) = \sum E(0,1,3,4,5,6,7,8,9). \tag{8}$$

Table 3. An example table.

No.	A	B	C	D	Action
E_0	0	1	0	1	Forward to 1
E_1	0	0	0	0	Forward to 1
E_2	0	0	1	0	Forward to 2
E_3	0	1	1	1	Forward to 1
E_4	0	1	0	0	Forward to 1
E_5	1	1	1	1	Forward to 1
E_6	1	0	1	0	Forward to 1
E_7	1	0	0	0	Forward to 1
E_8	1	0	1	1	Forward to 1
E_9	1	1	0	1	Forward to 1
E_{10}	1	0	0	1	Forward to 3

Step 1: find prime implicants (P_i). Here all minterms are placed in the minterm table as shown in Table 4, and Stage I is to combine the minterms. If two terms vary by only a single bit, that bit is replaced with a dash (-). Stage II is the result of Stage I, and Stage III is the result of combining the minterms in Stage II. '/' indicates if the entry is combined in the next stage.

Table 4. Minterm table.

Stage I				Stage II (Size 2 Implicants)			
No. of 1s	**$\Sigma(E_i)$**	**ABCD**	**P_i**	**No. of 1s**	**$\Sigma(E_i)$**	**ABCD**	**P_i**
0	1	0000	/	0	1, 4	0-00	P_7
					1, 7	-000	P_6
1	4	0100	/	1	4, 0	010-	P_5
	7	1000	/		7, 6	10-0	P_4
2	0	0101	/	2	0, 3	01-1	/
	6	1010	/		0, 9	-101	/
3	3	0111	/		6, 8	101-	P_3
	8	1011	/	3	3, 5	-111	/
	9	1101	/		8, 5	1-11	P_2
4	5	1111	/		9, 5	11-1	/
Stage III (Size 4 Implicants)							
No. of 1s		$\Sigma(E_i)$		ABCD		P_i	
2		0, 3, 9, 5		-1-1		P_1	

According to Table 4, all the prime implicants are shown as follows:

$$\begin{aligned}
P_1 &= \textstyle\sum E(0,3,9,5) = BD,\\
P_2 &= \textstyle\sum E(8,5) = ACD,\\
P_3 &= \textstyle\sum E(6,8) = AB'C,\\
P_4 &= \textstyle\sum E(7,6) = AB'D',\\
P_5 &= \textstyle\sum E(4,0) = A'BC',\\
P_6 &= \textstyle\sum E(1,7) = B'C'D',\\
P_7 &= \textstyle\sum E(1,4) = A'C'D'.
\end{aligned} \tag{9}$$

Step 2: find essential prime implicants (P_e).

From Table 4, none of the minterms can be combined any further. At this point, the table of essential prime implicant is constructed as in Table 5.

Table 5. The essential prime implicant table.

P_i	E_i								
	0	1	3	4	5	6	7	8	9
P_1	*		*		*				*
P_2					*			*	
P_3						*		*	
P_4						*	*		
P_5	*			*					
P_6		*					*		
P_7		*		*					

In order to find the essential prime implicants, each column needs to be checked whether there exists only one '*'. If a column has only one '*', the minterm can be covered by only one prime implicant. Then this prime implicant is essential. According to Table 5, P_1 is the only essential prime implicant.

As P_2 can be covered by P_1 and P_3, same as P_3, P_4, P_5, P_6, and P_7, they are not essential. In this example, the essential prime implicants cannot handle all the minterms (only E_0, E_3, E_5, and E_9 are covered). Therefore, other prime implicants are combined with P_1 to get the final result as:

$$F(A, B, C, D) = P_1 + P_2 + P_4 + P_5 = BD + ACD + AB'D' + A'BC'. \quad (10)$$

At last the final Table 6 is obtained as follows:

Table 6. The final table.

E_i	A	B	C	D	Action
E_{p1}	-	1	-	1	Forward to 1
E_{p2}	1	-	1	1	Forward to 1
E_{p3}	1	0	-	0	Forward to 1
E_{p4}	0	1	0	-	Forward to 1
E_2	0	0	1	0	Forward to 2
E_{10}	1	0	0	1	Forward to 3

The number of entries is aggregated from 11 to 6, and the compression ratio is 6/11 = 0.545.

3.3. Hidden Markov Model-Based Prediction

The goal of the proposed scheme based on HMM is to dynamically predict the popularity of the flow entries as accurately as possible and update the ExTable accordingly. After the flow entries are aggregated, the popularity of the flow entries are estimated periodically and the entries deemed to be popular are moved to ExTable. Note that the size of ExTable, n_p, is smaller than $(1 - C) \cdot N_{MFT}$ where C and N_{MFT} are the TCAM compression ratio and the size of entire MFT, respectively.

The match frequency of an entry indicates the popularity. For this, a counter, M, is associated to each flow entry, which is activated when the entry is installed in the flow table. M is the number of matches before the prediction occurs. Note that M may not be the only indicator of the popularity, and thus HMM is employed to estimate the probability of the flow entries to be matched in the near future.

The interarrival time of the flow is assumed to follow exponential distribution, and therefore the number of arrivals can be modeled using Poisson distribution. The probability of a flow arriving in

a given interval of time of Δt is predicted as follows. Assume that flow arrival occurs at any time. The probability of k arrivals in Δt is given by:

$$P(k,\ \Delta t) = \frac{e^{-\lambda \Delta t}(\lambda \Delta t)^k}{k!}. \tag{11}$$

The probability of at least one arrival in the interval Δt is given as:

$$P(k \geq 1,\ \Delta t) = 1 - e^{-\lambda \Delta t}. \tag{12}$$

The probability of a flow arriving in the next interval is computed as the mean value of the entire period from the beginning. It is:

$$P(k = 1,\ \Delta t) = e^{-\lambda \Delta t} \cdot (\lambda \Delta t). \tag{13}$$

HMM is effective to predict the probability of an observed sequence with the given triple $\lambda = (\pi, A, B)$. Let $H = \{H_0, H_1, \ldots, H_n\}$ be a set of hidden states, where H_i is defined as the number of time segments (Δt seconds per segment) a flow entry has not been matched from the initial state. For example, if there was no match for last $3\Delta t$ seconds, $H_3 = 3\Delta t$. If there was a match, $H_3 = 0$. Note that the hard timeout period (T_H) is preset, and an entry is forced to be evicted if no match occurs during T_H. Therefore, there will be n ($=T_H/\Delta_t$) segments before an entry is finally evicted, and H_n is the last state. Since $O = \{O_0, O_1, \ldots, O_n\}$ is a set of observable states of any entry, O_i indicates if the entry is matched in ith segment. It has two values; '1' for a successful match, '0' no match. Figure 4 shows the structure of the HMM of the proposed scheme.

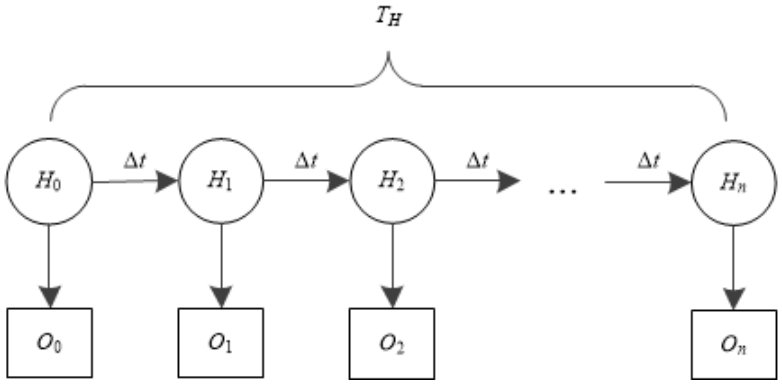

Figure 4. The state diagram of the proposed HMM.

With the HMM the probability of an observed sequence is found with the given parameters, A, B, and π. For a flow entry, there exist N ($=n + 1$) hidden states in its life time. Assume that there exist m time segments before the prediction occurs. Then the $(n - m) \times (n - m)$ hidden state transition probability matrix, A, and the $(n - m) \times 2$ emission probability matrix, B, are obtained as follows:

$$A = \begin{bmatrix} 1 - e^{-\lambda \Delta t} & e^{-\lambda \Delta t} & 0 & \cdots & 0 \\ 1 - e^{-\lambda \Delta t} & 0 & e^{-\lambda \Delta t} & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 - e^{-\lambda \Delta t} & 0 & 0 & \cdots & e^{-\lambda \Delta t} \end{bmatrix}, \tag{14}$$

$$B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{bmatrix}. \tag{15}$$

In order to compute the likelihood probability, $P(O|\lambda)$ of $O = \{O_0, O_1, \ldots, O_{(n-m)}\}$, the forward algorithm is adopted. Note that the probability of the observation sequence is obtained in which the value of O_i is 1 and the predicted last observable state is $O_{(n-m)}$. Then $P(O|\lambda)$ is calculated as follows.

1. Initialization. Each cell of the forward algorithm, $\alpha_t(j)$, represents the probability of hidden state H_j after checking the first t observations with the given λ. It expresses the probability as:

$$\alpha_t(j) = P(o_0, o_1, \cdots, o_t, H_t = q_j \big| \lambda). \tag{16}$$

Here $H_t = q_j$ denotes that the tth hidden state in the sequence is q_j. Then the initial probability is calculated as follows:

$$\alpha_0(j) = \pi_j b_j(o_0),\ 1 \le j \le (n-m). \tag{17}$$

If one or more matches occur during m segments, the following is obtained according to Equation (12) and the sum of π is 1.

$$\pi_0 = 1 - e^{-\lambda \Delta t}\ ,\ \pi_j = \frac{e^{-\lambda \Delta t}}{n-m} \tag{18}$$

If no match occurs, then:

$$\pi_0 = \frac{1 - e^{-\lambda \Delta t}}{(m+1)}\ ,\ \pi_j = \frac{e^{-\lambda \Delta t}}{(m+1)(n-m)}. \tag{19}$$

2. Recursion. For the hidden state sequence, H, and the observation sequence, O, the likelihood of O is estimated as:

$$P(O|H, \lambda) = \prod_{i=0}^{n-m} P(o_i \big| q_i) = \prod_{i=0}^{n-m} b_i(o_i). \tag{20}$$

While introducing π and A, it follows:

$$P(H|\lambda) = \prod_{i=0}^{n-m} P(q_i \big| q_{i-1}) = \pi_{q_0} \prod_{i=1}^{n-m} a_{q_{i-1} q_i}, \tag{21}$$

where q_0 is the initial state. Then the joint probability of H and O is:

$$P(O, H|\lambda) = P(O|H, \lambda) P(H|\lambda) = \prod_{i=0}^{n-m} b_i(o_i) \times \pi_{q_0} \prod_{i=1}^{n-m} a_{q_{i-1} q_i}. \tag{22}$$

Therefore, the total probability of the observations can be calculated by summing up all possible hidden state sequences:

$$P(O|\lambda) = \sum_H P(O, H|\lambda) = \sum_H [\prod_{i=0}^{n-m} b_i(o_i) \cdot \pi_{q_0} \prod_{j=1}^{n-m} a_{q_{j-1} q_j}]. \tag{23}$$

For each given state q_j at time t, the probability $\alpha_t(j)$ is estimated as:

$$\alpha_t(j) = b_j(O_t) \sum_{i=0}^{n-m} \alpha_{t-1}(i) a_{ij}, \tag{24}$$

where $0 \le j \le (n-m)$ and $1 \le t \le n$.

3. Termination. According to Equations (23) and (24), the probability of O is estimated as:

$$P(O|\lambda) = \sum_{j=0}^{n-m} \alpha_t(j). \tag{25}$$

Through the forward algorithm, the $P(O|\lambda)$ of each observed state can be computed. In order to decide the popularity of each entry, the value of O_i is set to 1 to record the probability of successful match.

Large $P(O|\lambda)$ means that the entry has high match probability. The number of popular flow entries selected in each flow table is denoted as k. Periodic ExTable update occurs in every ΔT. Here the popularity, ω, is decided based on the match frequency, M, and match probability, $P(O|\lambda)$, as follows:

$$\omega = \frac{M}{m} + P(O|\lambda). \tag{26}$$

After calculating the popularity, the flow entries of k largest popularity are moved to the ExTable. The number of flow entries in ExTable is $n_t \cdot k$ if there exist n_t flow tables ($n_t \cdot k \leq n_p$). If there exist several flow entries of the same value, the one of the longest remaining life time is selected. The proposed periodic entry selection scheme is depicted in Algorithm 2.

Algorithm 2. Selection operation of popular entry

Begin
create ExTable using the saved memory
set ΔT, k
  flowEntry.M[i][j] = 0
  flowEntry.P[i][j] = 0
  flowEntry.ω[i][j] = 0
while $0 < \Delta t < \Delta T$
  for i from 0 to n_t −1
    for j from 0 to n_f −1
      input O, λ = (π,A,B)
      initial $\alpha_0(j)$ by Equation (17), π by Equations (18) and (19)
      compute $\alpha_t(j)$ by Equation (24)
      calculate *flowEntry.P*[i][j] by Equation (25)
      count *flowEntry.M*[i][j]
      calculate *flowEntry.*ω[i][j] by Equation (26)
    end for
  end for
end while
if (Δt == ΔT)
  for i from 0 to n_t − 1
    sort *flowEntry.* ω[i][j] from large to small
  for j from 0 to k
    select *flowEntry*[i][j]
  end for
  end for
end if
update *flowEntry*[i][j] into ExTable
end

3.4. Flow Processing Time

The queuing model [19,37] of the proposed ExTable scheme is shown in Figure 5 where each of the nodes is considered as an M/M/1 queue. Table 7 is the list of variables used in the model.

Table 7. The variables used in the queueing model.

Variable	Description
n_t	Number of original flow tables
λ	Arrival rate of the flows
n_p	Number of flow entries in ExTable
t_f	Matching time with a flow entry
n_f	Number of flow entries in a flow table
t_e	Execution time of action with a packet
Rd_n	Rate of using direct path in each flow table
R_p	Match rate of the ExTable

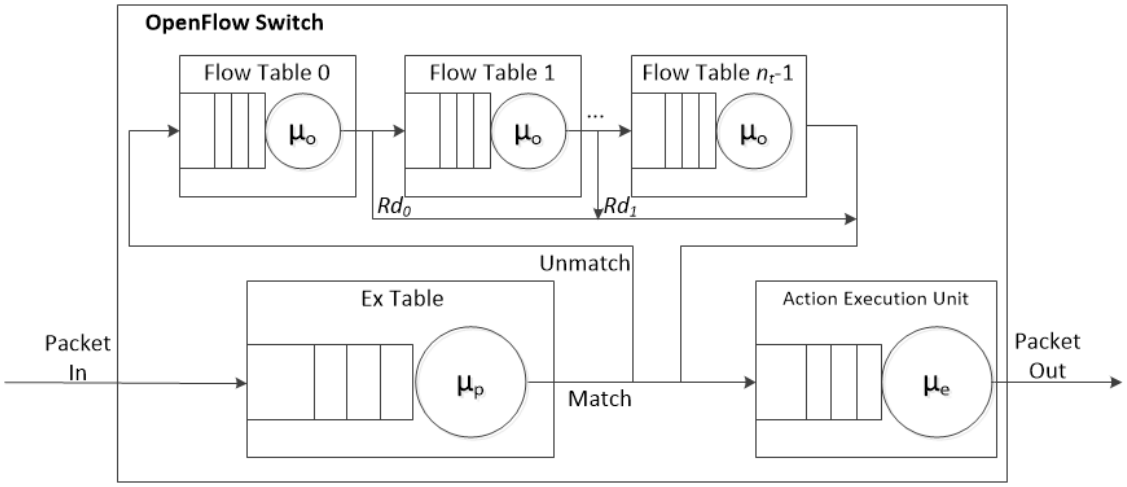

Figure 5. The queuing model of the proposed scheme.

According to Little's Law [37], the flow processing time in the system can be calculated as $T = N/\lambda$, where N is the average number of flows in the system and λ is the arrival rate of the flows. For obtaining the flow processing time of the proposed scheme, T_F, firstly the average number of flows in the system, N_F, needs to be obtained. In the following formula R_m is the match rate of the ExTable, and ρ_p, ρ_f, and ρ_e are the utilization of ExTable, flow table, and AEU, respectively. Note that there exists a direct path from each flow table to AEU. The rate of sending packets directly to AEU from the flow tables are $\{Rd_0, Rd_1, \ldots, Rd_{n_t-2}\}$, where $n_t \geq 2$. In order to calculate N_F, the average number of flows in ExTable, flow table 0, flow table 1, flow table (n_t-1) and AEU, N_p, N_{f0}, N_{f1}, $N_{f(n_t-1)}$, and N_e, should be estimated. They are calculated as follows:

$$N_p = \frac{\rho_p}{1-\rho_p} = \frac{\lambda n_p t_f}{1-\lambda n_p t_f}, \tag{27}$$

$$N_{f0} = \frac{\rho_{f0}}{1-\rho_{f0}} = \frac{(1-R_m)\lambda n_f t_f}{1-(1-R_m)\lambda n_f t_f}, \tag{28}$$

$$N_{f1} = \frac{\rho_{f1}}{1-\rho_{f1}} = \frac{(1-R_m-Rd_0)\lambda n_f t_f}{1-(1-R_m-Rd_0)\lambda n_f t_f}, \tag{29}$$

$$N_{f(n_t-1)} = \frac{\rho_{f(n_t-1)}}{1-\rho_{f(n_t-1)}} = \frac{\left[(1-R_m)-\sum_{i=0}^{n_t-2} Rd_i\right]\lambda n_f t_f}{1-\left[(1-R_m)-\sum_{i=0}^{n_t-2} Rd_i\right]\lambda n_f t_f}, \; n_t \geq 2, \tag{30}$$

$$N_e = \frac{\rho_e}{1-\rho_e} = \frac{\lambda t_e}{1-\lambda t_e}. \tag{31}$$

The average number of flows in the system, N_F, is as follows:

$$\begin{aligned} N_F = N_p + N_{f0} + \quad & N_{f1} + \cdots + N_{f(n_t-1)} + N_e \\ & = \lambda\Bigg(\frac{n_p t_f}{1-\lambda n_p t_f} + \frac{(1-R_m)n_f t_f}{1-(1-R_m)\lambda n_f t_f} + \frac{(1-R_m-Rd_0)n_f t_f}{1-(1-R_m-Rd_0)\lambda n_f t_f} + \cdots \\ & + \frac{\left[(1-R_m)-\sum_{i=0}^{n_t-2} Rd_i\right]n_f t_f}{1-\left[(1-R_m)-\sum_{i=0}^{n_t-2} Rd_i\right]\lambda n_f t_f} + \frac{t_e}{1-\lambda t_e}\Bigg). \end{aligned} \tag{32}$$

Note that there exist one ExTable, n_t flow tables and one AEU. As a result, the flow processing time, T_F, is obtained as follows:

$$\begin{aligned} T_F = \frac{n_p t_f}{1-\lambda n_p t_f} + \quad & \frac{(1-R_m)n_f t_f}{1-(1-R_m)\lambda n_f t_f} + \frac{(1-R_m-Rd_0)n_f t_f}{1-(1-R_m-Rd_0)\lambda n_f t_f} + \cdots \\ & + \frac{\left[(1-R_m)-\sum_{i=0}^{n_t-2} Rd_i\right]n_f t_f}{1-\left[(1-R_m)-\sum_{i=0}^{n_t-2} Rd_i\right]\lambda n_f t_f} + \frac{t_e}{1-\lambda t_e}. \end{aligned} \tag{33}$$

The queueing model of the flow processing time for ExTable is used later in computer simulation. The proposed scheme is simulated and compared with the existing schemes in the following section.

4. Performance Evaluation

In this section computer simulation is conducted to evaluate the TCAM compression ratio, prediction accuracy, and match rate of the proposed approach.

4.1. Simulation Environment

The simulation is conducted on Intel Core i5 process, 3.2 GHz PC with 8GB RAM, and Matlab R2014a. The flows used in the simulation are generated following exponential distribution with $\lambda = 1$. A virtual SDN environment is built with Floodlight controller, Open vSwitch, and end nodes emulated by Mininet. Here the Floodlight controller is linked to an Open vSwitch, and two end nodes are connected to the switch. The performance of the proposed scheme is compared with the original MFT approach and existing schemes [20,21].

For testing the proposed entry aggregation scheme, eight different numbers of flow entries are generated randomly. The number of entries is varied from 100 to 800. For obtaining the flow processing time for MFT and ExTable, the queueing models of them are used in the simulation. The number of flow entries in a flow table is set to 20, while the size of ExTable is $n_t \cdot k$. The service rate of the ExTable, flow table, and AEU are set to be 1.67, 1.67, and 10 [19,21], respectively. Tables 8 and 9 list the parameter values and the factors used in the simulation, respectively.

Table 8. The parameter setting of the simulation.

Parameter	Value
n_p	$n_t \cdot k$
n_f	20
μ_p	1.67
μ_f	1.67
μ_e	10

Table 9. The factors in the simulation.

Variable	Description
C	Compression ratio
k	Number of popular flow entries per table
n_t	Number of flow tables
λ	Flow arrival rate
t_h	Hard timeout value
R_d	Rate of using direct path in each flow table
ΔT	ExTable update period

The match rate and prediction accuracy of the proposed scheme are examined with various values of the operation parameters, and the flow processing time is compared with the existing schemes. Note that small processing time implies higher match rate and prediction accuracy.

The simulation is run 1000 times to achieve dependable result. The accuracy of the proposed HMM-based prediction is then calculated as:

$$Accuracy = \frac{N\_Success}{N\_Experiment} \times 100\%. \tag{34}$$

Here, *N_Success* indicates the number of selected flow entries actually having the largest number of matches, and *N_ Experiment* is the whole number of simulations. The match rate of the proposed ExTable scheme is obtained as follows:

$$Match\ rate = \frac{N\_Match}{N\_Flows} \times 100\%. \tag{35}$$

Here, *N_Match* is the number of incoming flows matching the ExTable, while *N_Flows* is the total number of incoming flows.

4.2. Simulation Results

Figure 6 shows the average compression ratios with eight different numbers of flow entries. In order to achieve dependable result, the simulation is run 1000 times for each number of flow entries generated randomly. Here compression ratio is the number of entries after reduction to that of original entries [8,32]. Therefore, lower compression ratio means more entries are aggregated. The pruning alone reduces the table size by almost 25%. The proposed scheme (pruning + QM) shows the best compression ratio among the four schemes compared.

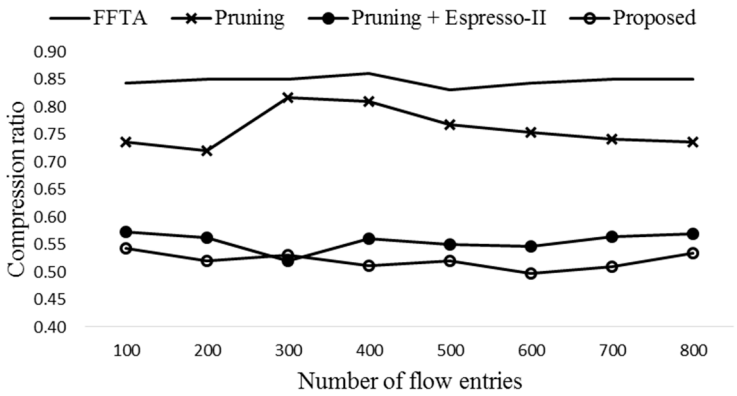

Figure 6. The compression ratios of four different schemes.

Figure 7 shows the prediction accuracy of the proposed HMM-based scheme with different sizes of flow table. The number of flows is 500 and ΔT (ExTable update period) is set to 20. In order to

see the effect of T_H, it is set to 50, 70, and 90. Observe that, as the number of entries of the flow table increases, the prediction accuracy increases. Also, the accuracy slightly decreases as T_H increases because many obsolete flow entries may remain in the flow tables as time goes by.

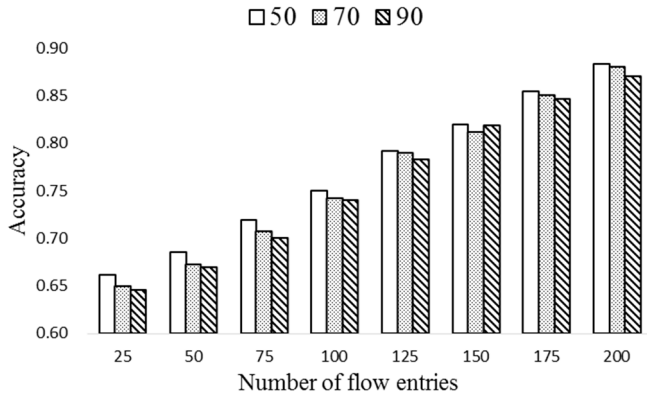

Figure 7. The prediction accuracy vs size of flow table.

Figure 8 compares the prediction accuracy of the proposed HMM-based scheme with the scheme based on only match frequency. Here the number of flows is 500 and (ΔT,T_H) are set to (20,70). Notice from the figure that the proposed scheme shows consistently higher accuracy, while the accuracy of both the schemes increases as the number of flow entries grows. Note that high match frequency may not be the only indicator of the popularity of an entry. Even though an entry has low match frequency, it could be a popular one if its match probability is high.

In Figure 9, the match rate of popular flow entries is obtained with different ΔT of 20, 40, and 60. According to the result of Figure 6, the space saving due to the compression is about 45%. Since $n_t \cdot k \leq 0.45 \cdot 20 n_t$, n_t and k are set to 5 and 9, respectively. Observe from the figure that the match rate decreases as ΔT increases, which demonstrates that ExTable needs to be updated as quickly as possible. The proposed scheme achieves nearly 68% match rate when ΔT is 20. As ΔT becomes large, some entries may not be popular any more. There exists a trade-off between the match rate and update cost.

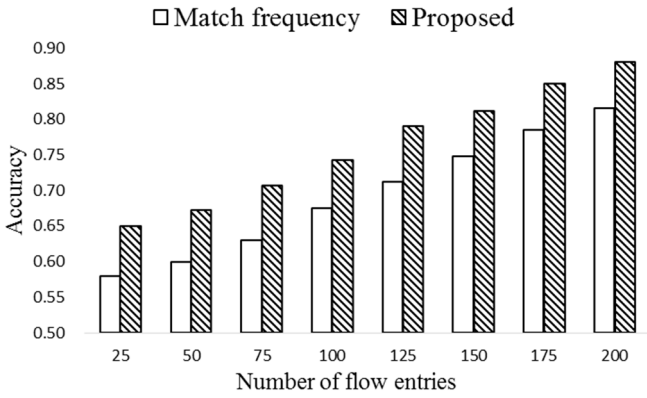

Figure 8. The comparison of prediction accuracies vs size of flow table.

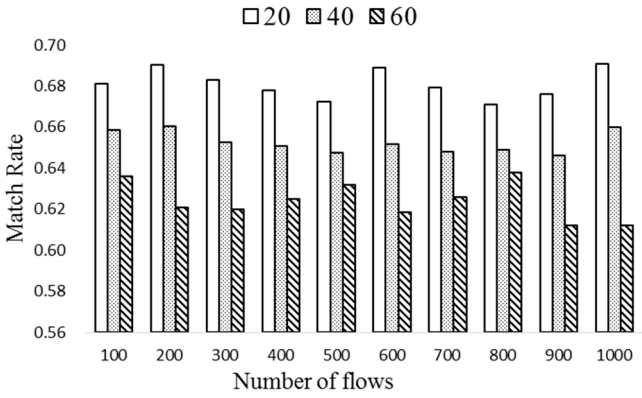

Figure 9. The match rates of ExTable with different ΔT.

Figure 10 evaluates the match rate of ExTable with different number of popular entries per flow table (k). Note that k can be varied from 1 to 9 each allowing different compression ratio. Here n_t is set to 5 and ΔT to 40, while k is varied from 3 to 9. Notice from Figure 10 that the match rate increases as k grows as expected. A proper number of popular entries needs to be selected which allows high match rate while requiring reasonable operation cost.

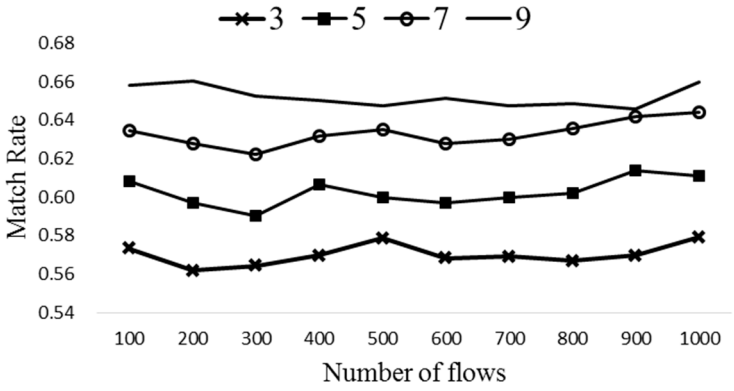

Figure 10. The match rates with different k.

In Figure 11, the match rate is obtained with different number of flow tables, n_t, of 3, 5, and 7. Here ΔT is 40 and k is 7. It is clear that the match rate of ExTable decreases as n_t grows since the number of total flow entries becomes larger. Therefore, k needs to be set properly considering various design parameters.

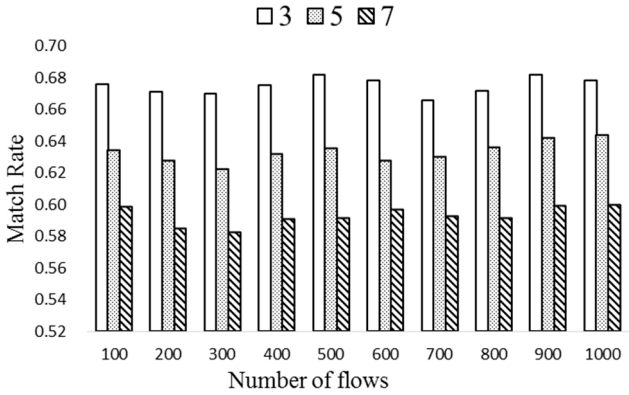

Figure 11. The match rates with different n_t.

Figure 12 compares the flow processing time of the three schemes, original MFT, MILE, and the proposed Agg-ExTable. Here (ΔT,R_m,n_p,n_f,k,n_t) of the proposed scheme are set to (20,0.68,35,20,7,5) using the data obtained from the simulation. The rate of using the direct path, Rd_n, is decided randomly between 0 and 0.1. Observe that the proposed Agg-ExTable scheme achieves much smaller flow processing time than the other schemes as the ExTable provides an express forwarding path for a large portion of incoming flows. The difference gets more significant as the flow arrival rate increases. Note that high arrival rate means high network load, and the flow processing time of all the schemes grows as the network load increases. Another important merit of the proposed scheme is that the processing time is almost constant regardless of the load unlike the other schemes. Figure 13 shows the flow processing times with different settings of (40,0.6,35,20,5,7). The proposed scheme consistently outperforms the other schemes.

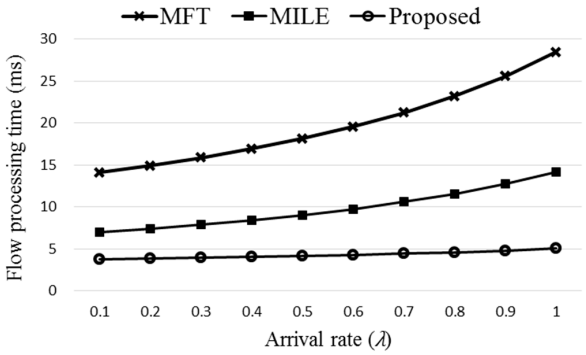

Figure 12. The comparison of flow processing times with varied arrival rate.

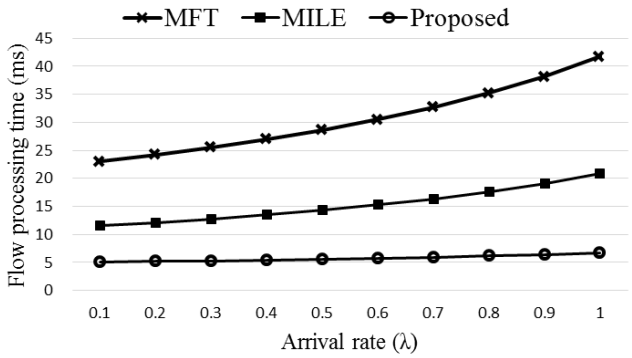

Figure 13. The comparison of flow processing times with varied arrival rate.

5. Conclusions

In this paper a novel flow management scheme has been proposed for MFTs of SDN switches. In the proposed Agg-ExTable scheme, the flow entries in the MFT are periodically aggregated by applying the pruning and Quine–Mccluskey algorithm. Utilizing the memory space saved by the aggregation, ExTable is constructed, keeping popular flow entries and allowing early match with the incoming flows. The proposed scheme is able to save about 45% TCAM space of MFT and efficiently decrease the flow processing time through the express forwarding path provided by the front-end ExTable. Popular flow entries are selected from the flow tables using the HMM, where popularity is decided based on the match frequency and match probability. Computer simulation revealed that the proposed scheme significantly outperforms the existing schemes in terms of flow processing time.

In the future, we plan to investigate the approach further reducing the memory space used for ExTable. Various parameters are involved in the proposed scheme. A formal model will be developed with which proper parameter values can be decided for the given condition. In addition, the match rate of the ExTable will be further improved with more sophisticated techniques such as machine

learning and fuzzy theory in the selection of popular entries. The proposed approach will also be tested with a real test bed for various operational conditions and SDN environments.

Author Contributions: Conceptualization, C.W.; Methodology, C.W. and H.Y.Y.; Software, C.W.; Validation, C.W., and H.Y.Y.; Formal Analysis, C.W.; Investigation, H.Y.Y.; Resources, H.Y.Y.; Data Curation, C.W.; Writing–Original Draft Preparation, C.W.; Writing–Review & Editing, H.Y.Y.; Visualization, C.W.; Supervision, H.Y.Y.; Project Administration, H.Y.Y.; Funding Acquisition, H.Y.Y.

Funding: This work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2016-0-00133, Research on Edge computing via collective intelligence of hyper-connection IoT nodes), Korea, under the National Program for Excellence in SW supervised by the IITP(Institute for Information & communications Technology Promotion)(2015-0-00914), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2016R1A6A3A11931385, Research of key technologies based on software defined wireless sensor network for real time public safety service, 2017R1A2B2009095, Research on SDN-based WSN Supporting Real-time Stream Data Processing and Multi-connectivity), the second Brain Korea 21 PLUS project.

Conflicts of Interest: The authors declare that there is no conflict of interest.

References

1. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
2. Kreutz, D.; Ramos, F.M.V.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
3. Stevens, M.; Ng, B.; Streader, D.; Welch, I. Global and local knowledge in SDN. In Proceedings of the 2015 International Telecommunication Networks and Applications Conference (ITNAC), Sydney, NSW, Australia, 18–20 November 2015.
4. Feamster, N.; Rexford, J.; Zegura, E. The road to SDN. *Queue* **2013**. [CrossRef]
5. Feamster, N.; Rexford, J.; Zegura, E. The road to SDN: an intellectual history of programmable networks. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–98. [CrossRef]
6. Open Networking Foundation, SDN in the Campus Environment. Available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-enterprise-campus.pdf (accessed on 3 April 2019).
7. Wang, S.; Li, D.; Xia, S. The problems and solutions of network update in SDN: A survey. In Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 26 April–1 May 2015.
8. Chao, T.-Y.; Wang, K.; Wang, L.; Lee, C.-W. In-switch dynamic flow aggregation in software defined networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017.
9. Yu, M.; Wundsam, A.; Raju, M. NOSIX: A lightweight portability layer for the SDN OS. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 28–35. [CrossRef]
10. Berde, P.; Gerola, M.; Hart, J.; Higuchi, T.; Kobayashi, M.; Koide, T.; Lantz, B.; O'Connor, B.; Radoslavov, P.; Snow, W.; et al. ONOS: Towards an open, distributed SDN OS. In Proceedings of the third workshop on Hot topics in software defined networking, Chicago, IL, USA, 22 August 2014.
11. Bannour, F.; Souihi, S.; Mellouk, A. Distributed SDN control: Survey, taxonomy, and challenges. *IEEE Commun. Surv. Tutorials* **2018**, *20*, 333–354. [CrossRef]
12. Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M.; et al. B4: Experience with a globally-deployed software defined wan. In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, Hong Kong, China, 12–16 August 2013.
13. Kempf, J.; Bellagamba, E.; Kern, A.; Jocha, D.; Takacs, A.; Skoldstrom, P. Scalable fault management for OpenFlow. In Proceedings of the 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012.
14. Scott-Hayward, S.; O'Callaghan, G.; Sezer, S. SDN security: A survey. In Proceedings of the 2013 IEEE SDN For Future Networks and Services (SDN4FNS), Trento, Italy, 11–13 November 2013.

15. Hu, Z.; Wang, M.; Yan, X.; Yin, Y.; Luo, Z. A comprehensive security architecture for SDN. In Proceedings of the 18th International Conference on Intelligence in Next Generation Networks, Paris, France, 17–19 February 2015.
16. Open Networking Foundation, Openflow Switch Specification Version 1.3.0. Available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf (accessed on 3 April 2019).
17. Open Networking Foundation, The Benefits of Multiple Flow Tables and Ttps. Available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_Multiple_Flow_Tables_and_TTPs.pdf (accessed on 3 April 2019).
18. Leng, B.; Huang, L.; Wang, X.; Xu, H.; Zhang, Y. A mechanism for reducing flow tables in software defined network. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015.
19. Ozcevik, Y.; Erel, M.; Canberk, B. Spatio-Temporal Multi-Stage OpenFlow Switch Model for Software Defined Cellular Networks. In Proceedings of the Vehicular Technology Conference (VTC Fall), Boston, MA, USA, 6–9 September 2015.
20. Liu, H. Reducing routing table size using ternary-cam. In Proceedings of the HOT 9 Interconnects. Symposium on High Performance Interconnects, Stanford, CA, USA, 22–24 August 2001.
21. Wu, Z.; Jiang, Y.; Yang, S. An Efficiency Pipeline Processing Approach for OpenFlow Switch. In Proceedings of the Local Computer Networks (LCN), Dubai, United Arab Emirates, 7–10 November 2016.
22. Iizawa, T. Fast tracker performance using the new "variable resolution associative memory" for ATLAS. In Proceedings of the Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), Anaheim, CA, USA, 27 October–3 November 2012.
23. Quine, W.V. The problem of simplifying truth functions. *Am. Math. Mon.* **1952**, *59*, 521–531. [CrossRef]
24. Quine, W.V. A way to simplify truth functions. *Am. Math. Mon.* **1955**, *62*, 627–631. [CrossRef]
25. McCluskey, E.J. Minimization of Boolean functions. *Bell Syst. Tech. J.* **1956**, *35*, 1417–1444. [CrossRef]
26. Baum, L.E.; Petrie, T. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Stat.* **1966**, *37*, 1554–1563. [CrossRef]
27. Baum, L.E.; Eagon, J.A. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Am. Math. Soc.* **1967**, *73*, 360–363. [CrossRef]
28. Baum, L.E.; Petrie, T.; Soules, G.; Weiss, N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.* **1970**, *41*, 164–171. [CrossRef]
29. Bevilacqua, V.; Daleno, D.; Cariello, L.; Mastronardi, G. Pseudo 2D Hidden Markov Models for face recognition using neural network coefficients. In Proceedings of the 2007 IEEE Workshop on Automatic Identification Advanced Technologies, Alghero, Italy, 7–8 June 2007.
30. Jurafsky, D.; Martin, J.H. Speech and Language Processing. Available online: http://www.cs.colorado.edu/~{}martin/SLP/Updates/1.pdf (accessed on 20 May 2019).
31. Liu, A.X.; Meiners, C.R.; Torng, E. TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs. *IEEE/ACM Trans. Networking (TON)* **2010**, *18*, 490–500. [CrossRef]
32. Luo, S.; Yu, H. Fast incremental flow table aggregation in SDN. In Proceedings of the Computer Communication and Networks (ICCCN), Shanghai, China, 4–7 August 2014.
33. Brayton, R.K.; Hachtel, G.D.; McMullen, C.; Sangiovanni-Vincentelli, A. *Logic Minimization Algorithms for VLSI Synthesis*; Springer Science & Business Media: Dordrecht, The Netherlands, 1984.
34. Karnaugh, M. The Map Method for Synthesis of Combinational Logic Circuits. *Trans. Am. Inst. Electr. Eng. Part I Commun. Elect.* **1953**. [CrossRef]
35. Rudell, R.L. Multiple-Valued Logic Minimization for PLA Synthesis. Available online: https://www2.eecs.berkeley.edu/Pubs/TechRpts/1986/ERL-86-65.pdf (accessed on 20 May 2019).
36. Holdsworth, B.; Woods, C. *Digital Logic Design*; Elsevier: Amsterdam, The Netherlands, 2002.
37. Adan, I.; Resing, J. *Queueing Theory*; Eindhoven University of Technology: Eindhoven, The Netherlands, 2002.