Adaptive Wildcard Rule Cache Management for Software-Defined Networks

MEDAPATI VENKATA KALYANI

PG Scholar, Department of Computer Science,

SVKP & Dr K S Raju Arts & Science College(A),
Penugonda, W.G.Dt., A.P, India
venkatakalyani5555@gmail.com

B.N.SRINIVASA GUPTA

Associate Professor in Computer Science,

SVKP & Dr K S Raju Arts & Science College(A),
Penugonda, W.G.Dt., A.P, India
bnsgupta@gmail.com

Abstract

Software-Defined Networking enables flexible flow control by caching rules at OpenFlow switches. Wildcard rule caching enables management of traffic aggregates, reduces flow setup queries, and simplifies policy management. However, to guarantee correct packet matching, some rules that depend on the requested rule need to be cached as well, which leads to unnecessary flow table bloat and potential overflow. We have proposed a scheme called CAching rules in Buckets (CAB) to mitigate the dependency issue by partitioning the field space into buckets and caching rules associated with the requested buckets. In this paper, we propose the Adaptive Cache ManagEment (ACME) for CAB, which dynamically adjusts the sizes and shapes of buckets according to incoming traffic to achieve more efficient flow table utilization. The improvement also includes preloading rules that span a wide field space to reduce bandwidth usage in the control channel. We formalize the caching policies for CAB-ACME to guarantee the semantic correctness of packet classification. We evaluate the performance of CAB-ACME through software-based simulations and a prototype built with the OpenDaylight controller and hardware switches from multiple vendors. The results show that, compared with other rule caching schemes, CAB-ACME reduces the cache miss rate by one order of magnitude and the control channel bandwidth usage by a half. ACME also helps maintain a steadier performance under dynamic traffic changes compared with the baseline CAB design.

Keywords : Control systems, Adaptive systems, Bandwidth, Adaptation models, Aerospace electronics, Memory management, Data centers.

1. INTRODUCTION

Software-defined networking is not a technology, but an architecture that provides support for virtual machine mobility independent of the physical network. The Open Networking Foundation (ONF) is the group that is most associated with the development and standardization of software-defined networks. According to the ONF, "Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow™ protocol is a foundational element for building SDN solutions."

According to the ONF, the SDN architecture is:

- **Directly programmable:** Network control is directly programmable because it is decoupled from forwarding functions.

- **Agile:** Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.

- **Centrally managed:** Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch

- **Programmatically configured:** SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software

- **Open standards-based and vendor-neutral:** When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN networking controllers instead of multiple, vendor-specific devices and protocols.
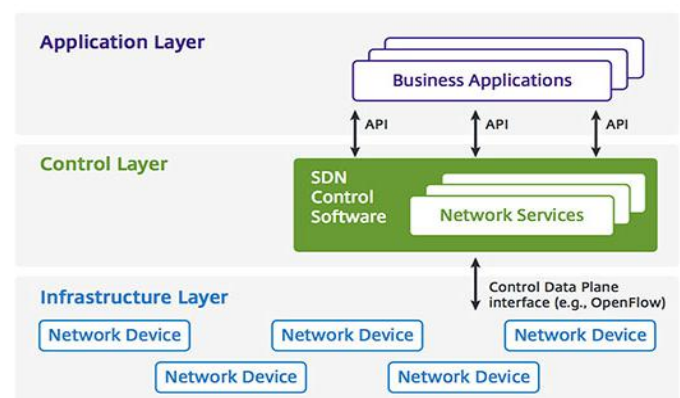

Figure 1 SDN System Architecture

Below is a description of some of the key concepts that are part of the SDN system architecture shown in Figure 1.

Business applications

This refers to applications that are directly consumable by end users. Possibilities include video conferencing, supply chain management and customer relationship management.

Network & security services

This refers to functionality that enables business applications to perform efficiently and securely. Possibilities include a wide range of L4 – L7 functionality including ADCs, WOCs and security capabilities such as firewalls, IDS/IPS and DDoS protection.

Pure SDN switch

In a pure SDN switch, all of the control functions of a traditional switch (i.e., routing protocols that are used to build forwarding information bases) are run in the central controller. The functionality in the switch is restricted entirely to the data plane.

Hybrid switch

In a hybrid switch, SDN technologies and traditional switching protocols run simultaneously. A network manager can configure the SDN controller to discover and control certain traffic flows while traditional,

distributed networking protocols continue to direct the rest of the traffic on the network.

Hybrid network

A hybrid network is a network in which traditional switches and SDN switches, whether they are pure SDN switches or hybrid switches, operate in the same environment.

Northbound API

Relative to Figure 1, the northbound API is the API that enables communications between the control layer and the business application layer. There is currently not a standards-based northbound API.

Southbound API

Relative to Figure 1, the southbound API is the API that enables communications between the control layer and the infrastructure layer. Protocols that can enable this communications include OpenFlow, the extensible messaging and presence protocol (XMPP) and the network configuration protocol.

Part of the confusion that surrounds SDN is that many vendors don't buy in totally to the ONF definition of SDN. For example, while some vendors are viewing OpenFlow as a foundational element of their SDN solutions, other vendors are taking a wait and see approach to OpenFlow. Another source of confusion is disagreement relative to what constitutes the infrastructure layer. To the ONF, the

infrastructure layer is a broad range of physical and virtual switches and routers. As described below, one of the current approaches to implementing network virtualization relies on an architecture that looks similar to the one shown in Figure 1, but which only includes virtual switches and routers.

In the rule caching system, different priorities are assigned to the rules to avoid conflicts because overlapping may exist in the field 'match' for the wildcard rules [11]. For the OpenFlow switch, the rule with highest priority will be applied first if there is a set of rules matching the packets in the cache. However, the rules dependency problem may cause issues within the wildcard rules caching system [15]. For example, if there are two overlapped wildcard rules, R1 and R2, whose field 'match' values are '000' and '00*' respectively. The input '*' represent the wildcard bit and it can be either '0' or '1'. That means R2 can match with '000' and '001', which overlaps with R1 in field 'match'. Suppose the priority of R1 is higher than R2. In TCAM, if we only cache the lower priority rule, the packets matching R1would instead match R2 and cause the incorrect matching problem. To solve this problem, extra space of cache need to be applied [15].

There are two kinds of dependency in wildcard rules, direct and indirect dependency. As for direct dependency, it means that there is an intersection between two wildcard rules in the field 'match'. When the rule is selected to be stored in TCAM, other rules with direct dependency to it are supposed to be cached in TCAM as well. The indirect dependency means that there are two rules depending on the same rule. These two rules have an indirect dependency even though they do not intersect with each other in the field 'macth'.

In the paper [15], the wildcard rule caching algorithm is using the layer-by-layer calculation based on the rule dependency DAG. There are two sub algorithms. The first is to calculate the accumulative contribution for one given un-cached rule and select the combination of rules with maximal contribution value. The second part is the overall wildcard rules caching algorithm achieved by applying the first sub-algorithm.

In the first sub-algorithm, it forms the new layer with the rules dependent on the selected rules. In the layer, each rule is going to be added into the selected rules set to calculate the accumulative contribution. The order of the rules to be added is dependent on their individual contribution, which is ranked from the highest to the lowest. After adding all candidate rules on the same layer to the rule set, the algorithm will select the combination of rules with highest accumulative contribution value. Next, the algorithm will repeat the process until there are not enough TCAM entries to cache.

From the wildcard rule caching algorithm [15], the layer -by-layer calculation would not give the combination of rules with highest accumulative contribution value in some cases, because this algorithm would use up the available TCAM entries before it reaches the rules with high contribution but on the higher level. Besides, the algorithm uses three constants to limit the number of candidate rules in each layer, the number of layers and the amount of chosen un-cached rules respectively. These constants would affect the result and should be modified to fit different input rules set.

This work presented the preliminary design of a novel reactive wildcard rule caching system named CAB in [13]. The main idea of CAB is to partition the geometric representation of the rule set, or the *field space* of packet headers, into many small logical structures named *buckets*, and to associate each bucket with one or multiple rules according to its location in the field space. Using the flow table pipeline supported by OpenFlow switches, buckets work as filters for packets to match the rules. By caching the bucket with associated rules, CAB is able to ensure correct packet forwarding.

2. LITERATURE REVIEW

Sanghyeon Baeg [1] 2008, Power consumption is the most critical issue for low-power ternary content-addressable memory (TCAM) in match lines designs. In the proposed match-line architecture, the match line present in each TCAM word is partitioned into four segments and is selectively pre-charged to reduce the match-line power consumption. The match lines which are partially charged are evaluated to determine the final comparison result by sharing the charges deposited in various parts of the partitioned segments.

B. Heller et al, [2] 2010, Built ElasticTree, which through data-center-wide traffic management and control, introduces energy proportionality in today's non-energy proportional networks. They will likely essentially decrease this quickly developing vitality cost. Compare multiple strategies for finding the minimum-power network [20]. The framework is vitality proficiency, best execution, and adaptation to non-critical failure. The system worked near its ability will build the possibility of dropped and postponed bundles.

A.R. Curtis et al, [3] 2011, DevoFlow proposition enables administrators to target just the streams that issue for their administration issue. DevoFlow handles most miniaturized scale streams in the information plane and consequently enables us to make the most out of switch resources. DevoFlow takes care of the issue by permitting a clonable trump card principle to choose a yield port. Multipath steering to statically stack balance movement with no utilization of the control-plane. These procedures don't spare much vitality on elite systems.

P. Porraset al, [4] 2012, Incorporates several critical components that are necessary for enabling security applications in Open Flow networks including role-based authorization, rule reduction, conflict evaluation, and policy synchronization. FortNOX is a critical initial phase in enhancing the security of Open Flow systems. It shows the achievability and suitability of our nom de plume set guideline decrease approach [18]. It is unable to handle the dynamic matching process.

Zahid Ullah et al, [5] 2012, Hybrid partitioned static random is a memory architecture in which access memory-based ternary content addressable memory (HP SRAM-based TCAM), which involves TCAM functionality with conventional SRAM, where we are eliminating the inherited disadvantages of conventional TCAMs. HP SRAM-based TCAM is a technique in which they logically dissect conventional TCAM table in a hybrid way (column-wise and row-wise) into TCAM sub-tables, which are then processed to be mapped to their corresponding SRAM memory units.

H. Kim and N. Feamster et al, [6] 2013, Designed and implemented Procera, an event-driven network control framework based on SDN. Additionally, utilize the OpenFlow convention to impart between the Procera controller and the hidden system switches. It gives better permeability and command over undertakings for performing system. This SDN can improve common network management tasks [19]. Procera experiences the characteristic deferral caused by the communication of the control plane and the information plane.

M. Yu, L. Jose et al, [7] 2013, OpenSketch empowers a straightforward and proficient approach to gather estimation information. It utilizes information plane estimation natives dependent on ware switches and an adaptable control plane so administrators can without much of a stretch execute variable estimation calculations. It has a simple, efficient way to control switches [16]. Sketches more flexible in supporting various measurement tasks. Delay of each measurement pipeline component is large.

Weirong Jiang et al, [8] 2013, Random access memory i.e. (RAM)-based Ternary Content Addressable Memory i.e.(TCAM) architecture is design for efficient implementation on state-of-the-art FPGAs. We give a formal study on RAM-based TCAM to disclose the ideas and the algorithms behind it. To face the timing challenge, we propose a modular architecture consisting of arrays of small-size RAM-based TCAM units.

Jacobson et al, [9] 2014, Novel control plane architecture called OpenNF that addresses these challenges through careful API design. OpenNF enables applications to settle on reasonable decisions in meeting their destinations. NF software is always Up-to-Date.

The system has High performance on network monitoring.

M. Moshref et al, [10] 2014, DREAM enables operators and cloud tenants to flexibly specify their measurement tasks in a network and dynamically allocates TCAM resources to these tasks based on the resource-accuracy. User-specified high level of accuracy. DREAM can support more concurrent tasks. DREAM needs to dismiss almost half of the assignments and drop about 10%.

N. Katta et al, [11] 2014, CacheFlow system is a system which "caches" the most popular rules in the small TCAM, in which they are relying on software to handle the small amount of "cache miss" traffic. But, we cannot blindly apply existing cache-replacement algorithms, because of dependencies between rules with overlapping patterns.

Naga Katta et al, [12] 2014, Instead of creating long dependency chains to cache smaller groups of rules in which semantics of the network policy are preserved. There are mainly four types of criteria for it. Elasticity which combines the best of hardware and software switches. Transparency which faithfully supporting native OpenFlow semantics, including traffic counters. Fine-grained rule caching which places popular rules in the TCAM, despite dependencies on less-popular rules. Adaptability which enables incremental

changes to the rule caching as the policy changes.

Bo Yan, Yang Xu, Hongya et al, [13] 2014, propose a wildcard rule caching system for SDN named CAching in Buckets (CAB). CAB is designed to partition the field space into logical structures called buckets. That buckets are consist of all the associated rules. By using CAB, we resolve the rule dependency problem along with small storage overhead. While Comparing to previous schemes, CAB reduces the low setup requests by making use of the order of magnitude, again saves control bandwidth by half, and significantly reduce average low setup time.

R. Wei et al. [14] 2016, Propose a new technique called Block Permutation (BP) to reduce the number of TCAM entries required to represent a classifier. The BP procedure altogether enhances the pressure rate the situation being what it is [17]. They pack the parcel grouping rules put away in TCAMs. TCAM can likewise be connected to other equipment usage based applications.

3. Existing System

In existing, TCAM Razor, DomainFlow and Palette algorithms are used. TCAM Razor is a compression algorithm based on prefix classifiers. TCAM Razor represents prefix classifiers to wildcard formation, which puts the

wildcards at the end of classifier. Then, it decomposes the multidimensional classifiers into one-dimensional classifiers and solve one-dimensional classifiers optimization problem.

DomainFlow separates a data center topology into three parts, Domain 1, the turning point, and Domain 2. Domain 1 is the flow control with wildcard matching table and Domain 2 is the flow control with exact matching table. As a result, all rules can be partitioned into the wildcard matching rules and the exact matching rules, which will be assigned to Domain 1 and Domain 2, respectively.

Palette is a distributing framework using Pivot Bit Decomposition (PBD) iteratively to decouple the table in order to preserve the correct semantics. PBD selects one pivot bit in the table and splits the table into two smaller sub-tables at all iteration. Then, they tried to let each packet traverses all the sub-tables in order to fulfill semantically equivalent according to the original table.

3.1 Disadvantages

These existing works provides poor caching ratio and less hit ratio

4. Proposed System

To deal with existing disadvantages, this work proposed a novel wildcard-rule caching algorithm and a cache replacement algorithm to make use of TCAM space efficiently. TCAM can look up a packet's header and compare the matching patterns of the packet to the match field of all rules in the flow table in parallel. Our wildcard-rule caching algorithm repeats caching a set of important rules into TCAM until there is no TCAM space. Our cache replacement algorithm takes temporal and spatial traffic localities into consideration, which could make hit ratio high.

4.1 Advantages

The proposed wildcard-rule caching algorithm could have better caching ability than the other existing algorithms. Furthermore, the proposed cache replacement algorithm could have higher hit ratio than the other existing algorithms.

5. IMPLEMENTATION

5.1 Bucket Generation

In this module, the initial bucket is generated based on the rule set. It requires each bucket to contain no more than N associated rules, where N is a pre-determined parameter. According to CAB's rule caching policy, the maximum number of entries that are installed or replaced in the switch for any individual query is bounded by $N + 1$.

The initial bucket set is generated by repeatedly partitioning the field space into smaller hyper-rectangles. The bucket tree records the process to partition the entire field space into smaller hyper-rectangles, and finally into buckets.

5.2 Rule Preloading

This module preloads the large rules and excludes them from the reactive rule caching process. A rule to be 'larger' when it is associated with a larger number of buckets. By sorting the rules by the number of associated buckets, the top K rules are identified as large rules to be preloaded. The amount of memory reserved for rule preloading or the setting of K affects cache performance. With more rules being preloaded, the control bandwidth usage by rule caching tends to be smaller. However, less flow table space is available for rules to be cached on demand, and flow table overload is more likely to happen. If we reserve less memory for rule preloading, the system reacts better to bursty flow arrivals, while the control channel load tends to be higher. Essentially the choice of K reflects a trade-off between the tolerance of traffic burstiness and the control channel load.
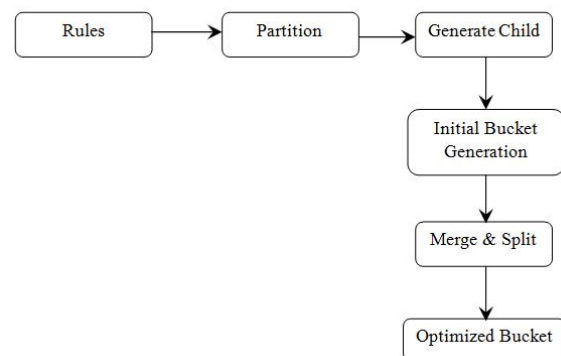
5.3 Bucket Adjustment

This module adjusts buckets with traffic-awareness that saves table memory in two ways. Using Optimized buckets reduces the number of buckets that need to be cached. The number of rules being cached but not matched by any packets is less in the optimized case. Each bucket adjustment is a series of merging operations followed by a series of splitting operations.
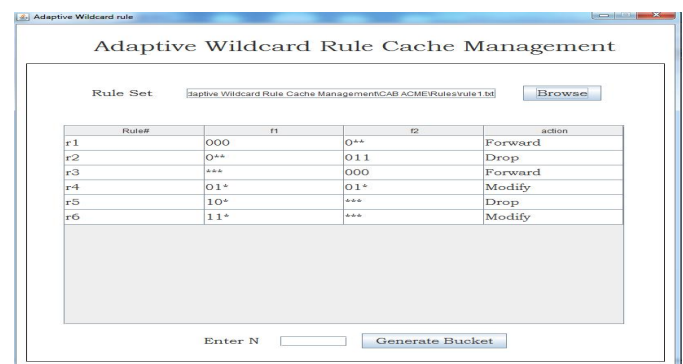
5.4 Rule Cache and Replacement

In the switch, each bucket is cached as a wildcard flow entry and can be stored in a *bucket filter*, while the rules are kept in the *rule table*. The caching is triggered when the switch receives *Flow-Mod* messages carrying corresponding buckets or rules. Evict inactive entries and save memory for new entries. The timeout mechanism is used to evict inactive entries. Whenever a bucket is installed, the CAB controller assigns the same hard timeouts for both the bucket and its associated rules.
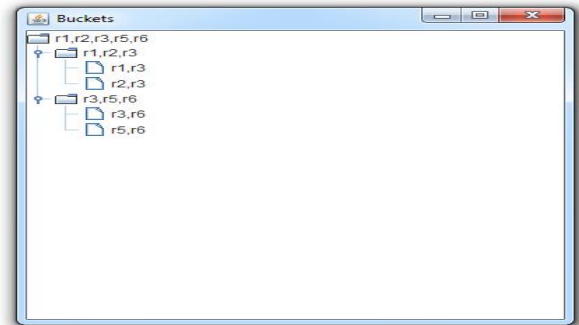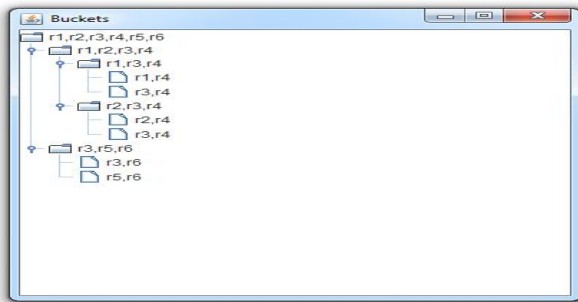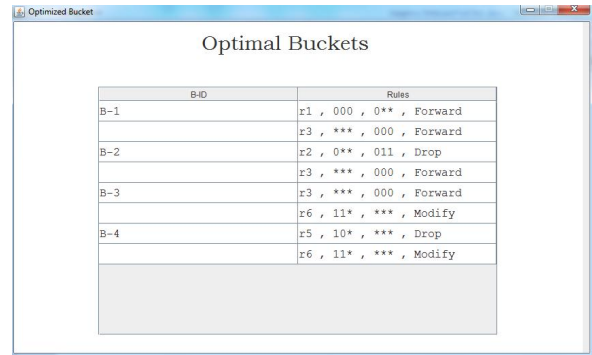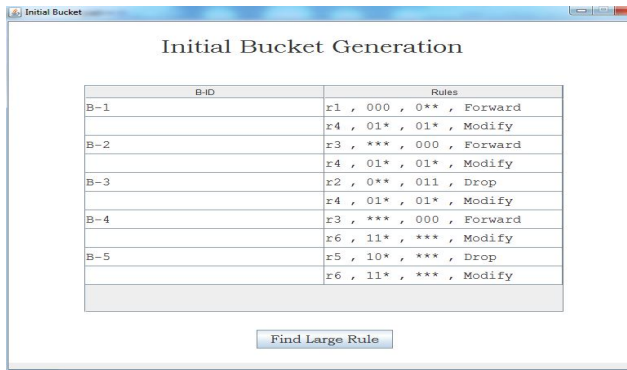
6. SYSTEM ARCHITECTURE

.

7. OUTPUT RESULTS

Adaptive Wildcard Rule Cache Management

Rule Set: daptive Wildcard Rule Cache Management\CAB ACME\Rules\rule1.txt   Browse

Rule#	f1	f2	action
r1	000	0**	Forward
r2	0**	011	Drop
r3	***	000	Forward
r4	01*	01*	Modify
r5	10*	***	Drop
r6	11*	***	Modify

Enter N 2   Generate Bucket

Pre and Post Order Traversal

Pre-Order	Post-Order
r1,r2,r3,r4,r5,r6	r1,r4
r1,r2,r3,r4	r3,r4
r1,r3,r4	r1,r3,r4
r1,r4	r2,r4
r3,r4	r3,r4
r2,r3,r4	r2,r3,r4
r2,r4	r1,r2,r3,r4
r3,r4	r3,r6
r3,r5,r6	r5,r6
r3,r6	r3,r5,r6
r5,r6	r1,r2,r3,r4,r5,r6

Initial Bucket Generation

B-ID	Rules
B-1	r1 , 000 , 0** , Forward
	r4 , 01* , 01* , Modify
B-2	r3 , *** , 000 , Forward
	r4 , 01* , 01* , Modify
B-3	r2 , 0** , 011 , Drop
	r4 , 01* , 01* , Modify
B-4	r3 , *** , 000 , Forward
	r6 , 11* , *** , Modify
B-5	r5 , 10* , *** , Drop
	r6 , 11* , *** , Modify

Find Large Rule

Optimal Buckets

B-ID	Rules
B-1	r1 , 000 , 0** , Forward
	r3 , *** , 000 , Forward
B-2	r2 , 0** , 011 , Drop
	r3 , *** , 000 , Forward
B-3	r3 , *** , 000 , Forward
	r6 , 11* , *** , Modify
B-4	r5 , 10* , *** , Drop
	r6 , 11* , *** , Modify

Buckets
- r1,r2,r3,r5,r6
  - r1,r2,r3
    - r1,r3
    - r2,r3
  - r3,r5,r6
    - r3,r6
    - r5,r6

Buckets
- r1,r2,r3,r4,r5,r6
  - r1,r2,r3,r4
    - r1,r3,r4
      - r1,r4
      - r3,r4
    - r2,r3,r4
      - r2,r4
      - r3,r4
  - r3,r5,r6
    - r3,r6
    - r5,r6

Large Rule Preloading

r4   01*   01*   Modify

Optimial Bucket

7. CONCLUSION

This work present the design and implementation of a novel wildcard rule caching system named CAB-ACME. CAB guarantees correct packet forwarding by partitioning the field space into buckets and caching buckets along with all the associated rules. ACME dynamically optimizes the buckets to accommodate traffic dynamics, which improves

flow table efficiency and reduces control channel loads. Through extensive simulations and prototype experiments, we validate the performance improvements of CAB-ACME over traditional rule caching schemes, and prove the feasibility of deploying CAB-ACME on commodity switches.

8. REFERENCE

[1] Sanghyeon Baeg,"Low-Power Ternary Content-Addressable Memory Design Using a Segmented Match Line, IEEE Transactions On Circuits And Systems—I: Regular Papers, Vol. 55, No. 6, July 2008.

[2] B. Heller et al., "Elastictree: Saving energy in data center networks", in Proc. NSDI, vol. 3. 2010, pp. 19–21.

[3] A.R. Curtis, "DevoFlow: Scaling flow management for high-performance networks", Comput. Commun. Rev., vol. 41, no. 4, pp. 254–265, Aug. 2011.

[4] P. Porraset al,"A security enforcement kernel for OpenFlow networks", in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 121–126.

[5] Zahid Ullah, Kim Ilgon, and Sanghyeon Baeg,"Hybrid Partitioned SRAM-Based Ternary Content Addressable Memory", IEEE Transactions On Circuits And Systems—I: Regular Papers, Vol. 59, No. 12, December 2012.

[6] H. Kim and N. Feamster,"Improving network management with software-defined networking", IEEE Commun. Mag., vol. 51, no. 2, pp. 114–119, Feb. 2013.

[7] M. Yu, L. Jose, and R. Miao, "Software-defined traffic measurement with OpenSketch", in Proc. NSDI, vol. 13. 2013, pp. 29–42

[8] Weirong Jiang, "Scalable Ternary Content Addressable Memory Implementation Using FPGAs", IEEE Architectures for Networking and Communications Systems December 2013.

[9] A. Gember - Jacobson et al,"OpenNF: Enabling innovation in network function control", in Proc. ACM Conf. SIGCOMM, 2014, pp. 163–174.

[10] M. Moshref, M. Yu, R. Govindan, and A. Vahdat,"DREAM: Dynamic resource allocation for software-defined measurement, in Proc. ACMConf. SIGCOMM, 2014, pp. 419–430.

[11] N. Katta, O. Alipourfard, J. Rexford, and D. Walker,"Rule-Caching Algorithms for Software-Defined Networks", available online at https://www.cs.   princeton.edu/~jrex/ papers/cacheflow-long14.pdf, 2014.

[12] Naga Katta, Omid Alipourfard, Jennifer Rexford and David Walker,"Infinite CacheFlow in Software-Defined Networks", ISBN: 978-1-4503-2989-7, August 2014.

[13] Bo Yan, Yang Xu, Hongya Xing, Kang Xi, H. Jonathan Chao,"CAB: A Reactive Wildcard Rule Caching System for Software-Defined Networks", Association for Computing Machinery, August 2014.

[14] R. Wei, Y. Xu, and H. J. Chao,"Finding non-equivalent classifiers in Boolean space to

reduce TCAM usage", IEEE/ACM Trans. New., vol. 24, no. 2, pp. 968–981, Apr. 2016.

[15] J. Sheu, and Y. Chuo, "Wildcard Rules Caching and Cache Replacement algorithm in Software-Defined Networking", IEEE Trans. On Networking and Service Management, pp. 19-29, Hsinchu, Taiwan, March 2016.

[16] Q. Dong, S. Banerjee, J. Wang, and D. Agrawal,"Wire-speed packet classification without TCAMs: A few more Registers.

ABOUT AUTHORS:

MEDAPATI VENKATA KALYANI is currently pursuing MCA in SVKP & Dr K S Raju Arts & Science College(A), affiliated to Adikavi Nannaya University, Rajamahendravaram. Her research interests include Operation Research, Design and Analysis of Algorithm and Big Data.

B.N.Srinivasa Gupta is working as Associate Professor in SVKP & Dr K S Raju Arts & Science College(A), Penugonda, A.P. He received Masters Degree in Computer Applications from Andhra University and computer Science & Engineering from Jawaharlal Nehru Technological University (JNTUK), kakinada, India. His research interests include Data Mining, Cyber Security, Artificial Intelligence.