A Popularity-Based Caching Strategy for Improved Efficiency in SVRF-based Multicast Control-Planes

Ruisi Wu and Wen-Kang Jia

***Abstract*—The packet forwarding engine (PFE), a vital component of high-performance switches and routers, plays a pivotal role in rapidly selecting the appropriate output port for tens of thousands of packets. The performance of the PFE hinges on the efficacy of the group membership algorithm. In this research, we present a hybrid approach called caching scalar-pair and vectors routing and forwarding (CSVRF), which comprises virtual output port bitmap caching (VOPBC) and fractional-*N* SVRF, designed to address significant multicast forwarding challenges such as scalability. We achieve this through the implementation of content address memory (CAM). Within the CSVRF framework, we introduce an innovative virtual output port bitmap cache table, which encompasses the most frequently occurring combinations of output port bitmaps (OPB). Furthermore, we divide the larger scalar-pair into *N* subgroups to enhance the reusability of prime resources. We validate our findings using Matlab-based mathematical models and simulations. Our results demonstrate significant decreases in both memory space usage and forwarding latency. Our approach assures minimized memory consumption, faster processing, and robust scalability in high port-density settings.**

***Index Terms*—Membership querying, multicast, packet forwarding engine (PFE), scalar-pair vectors routing and forwarding (SVRF).**

I. Introduction

MULTICAST communications increasingly attracts attention for scalable and efficient data delivery in the networking industry. Due to the growing demands of computing power such as that in cloud computing and datacenters, the major network equipment–switches and routers must be scalable and efficient for connecting tens or even hundreds of thousands of subordinate end hosts to superordinate network devices.

The performance of today's switches and routers is constrained by the interconnection technology. An essential and most complex component in a large-scale packet switching equipment such as high-end Ethernet switches and core IP routers is the hardware-assisted *packet forwarding engine (PFE)*, which interconnects all of the network traffic crossover, and holds a pivotal role to satisfy the exchange requirements of the Internet-scale traffic [1], [2]. The function of a PFE is forwarding packets as quick as possible from the input queue to the output queue through a backplane (i.e., switch fabric) of the data-plane. If there is no matched entry in the forwarding table to find the egress port(s) of the packet, the PFE drops the packet with an unknown destination and informs the routing engine for appropriate processing [3]. The control-plane of PFE determines and instructs the forwarding egress(es) of the data-plane through the forwarding table (a.k.a. forwarding information base (FIB)), which is built up by the upper-layer routing engine (RE) based on dynamic routing protocols and routing table (a.k.a. routing information base (RIB)). Since it requires huge system resources, only software implementation is feasible.

Different from the unicast routing and forwarding, mainstream multicast routing protocols such as MOSPF [4], PIM [5], DVMRP [6], and CBT [7] need a very large size of memory to store a huge number of states of the multicast routing table (MRT) in the multicast-enabled PFEs. Generally, MRT is not directly utilized for PFEs in modern switch/router architectures, instead it is usually transformed into smaller multicast forwarding tables (MFT), which consist of a set of multicast forwarding entries (MFEs). Each MFE contains only one unique flow identifier, output port identifier(s), and corresponding next-hop node identifier(s). If necessary, all of them are chosen by the RE as preferred routes for packet forwarding, and the MFT is directly used by a faster control-plane of PFE to control the forwarding of multicast packet flows along the on-tree switches/routers [3].

Each multicast group has more complicated forwarding information including its own specific branch pattern (i.e., indicating that it has multiple egress ports), which cannot be simply aggregated as in the unicast [8]–[10]. We could easily figure out that a certain number of multicast flows may consume much more resources including memory space and computing power than that consumed by the same number of unicast flows. The memory resource of MRT/MFT in on-tree switches/routers may be filled up quickly when a certain number of large multicast groups and/or a large number of small multicast groups are launched in the network. As predefined memory space of a MFT might be very restricted due to the expected cost and scale, and the maximum number of multicast groups supported on a PFE will be strictly adhered to. It will become a non-scalable issue and may frequently occur at each on-tree switch/router located near the multicast traffic hub. This issue can be treated as a structural limitation inherent in today's Internet, and an insuperable bottleneck in the large-scale multicast-enabled networks [11]–[15].

Manuscript received March 17, 2023; revised August 24, 2023; approved for publication by Seeling, Patrick, Division 3 Editor, October 27, 2023.

This research was sponsored by the National Natural Science Foundation of China (Project No. 61871131).

R. Wu is with the College of Photonic and Electronic Engineering, Fujian Normal University, Fuzhou, Fujian, China, and Graduate School of Information Science and Technology, Osaka University, Suita, Osaka, Japan, email: qsx20200789@student.fjnu.edu.cn, r-wu@ist.osaka-u.ac.jp.

W.-K. Jia is with the College of Photonic and Electronic Engineering, Fujian Normal University, Fuzhou, Fujian, China, email: wkjia@fjnu.edu.cn.

W.-K. Jia is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2023.000054

1229-2370/24/$10.00 © 2024 KICS

The group membership algorithm is an essential building block in the packet forwarding function of PFEs, it provides lookup processes in a group with a consistent view of the group membership. In a unicast packet, the network-layer destination address is used by a group membership algorithm to find a particular entry in the forwarding table, while in a multicast packet, a multicast address is used by a multiset group membership algorithm to find a single particular entry in the multicast forwarding table. If the membership is not found for a multicast packet in a group membership query, the packet is considered invalid and dropped in the input queue, or an exception handler such as PACK_IN in software-defined networks (SDNs) is executed. For unicast forwarding, the group membership query responded by a PFE to forward the packet is a particular output interface identifier (a.k.a. an egress) and a corresponding next-hop address if necessary. While in multicast forwarding, the result from a multiset group membership query is a set of egresses. In computation theory, the group membership algorithm transforms and simplifies the query question: "Where should this packet go?" into the true-false question: "Whether this packet should be forwarded to this egress?", which enables a PFE to make decisions from a massive number of forwarding elements (entries) within ultra-low processing latency. Nevertheless, due to complicated membership queries upon massive membership, it results in serious performance degradation, and conventional multicast forwarding algorithms are unable to accommodate such a high-intensity task, including high port-density (a.k.a. a large number of egresses) and massive membership-capacity (a.k.a. a huge number of forwarding entries) features [12], [16], [17].

To accommodate the scalability problems of managing a massive number of large-scale groups, quite a few group membership querying algorithms have been proposed in the past two decades. Most of these proposals tried to reduce memory consumptions by using *bloom filter (BF)* [18], which has indeed attracted much attention in the research community of both computation and networking recently [19]. Unfortunately, the existing BFs are neither effective in multiset group membership query for dynamic and rapid changes in group members, nor sufficient for handling false-positive problems [20]. So far, the Internet-scale multicast solution is still an unpopular research area in both academics and industries and only a few novel schemes have been conducted. However, with the progress of the market trend, launching a native multicast-enabled network technology such as IPv6 is imminent. We have observed that carrier-grade multicast-enabled PFE design is facing new challenges, hence it is a promising area that deserves considerable effort in investigation.

In this paper, we assume the selection of an output port is subject to the power-law distribution, in particular, Zipf distribution, also known as the 80/20 rule. The egress port, serving as the output interface of a router/switch for the transmission of packets to subsequent network nodes, assumes a critical role in network communication. A notable phenomenon arises in the selection of output ports, governed by a power-law distribution. In this case, where the majority multicast packets preferentially designate a few available ports. And the remaining packets will distribute themselves across the residual ports, which means that there is an uneven distribution in output port selection. Meanwhile, the ratio between is subject to Zipf distribution. For instances if a flow which contain 1000 packets arrived in an eight-port router/switch, the influence of the power-law distribution governing the selection of output ports can be showed as that a distinctive pattern emerges that most packets primarily opt for ρ_1 and ρ_2 as their primary output ports, and a few additional ports. On contrast, residue packets will select ρ_3, ρ_4, $\cdots$, and ρ_8 as their mainly output ports with a limited number of ρ_1 and ρ_2, adhering to the principles outlined by the Zipf distribution. And here ρ means the corresponding port in the router/switch. In such a distinctive pattern, a considerable repetition of output port combinations was observed in the forwarding table such as ρ_1 and ρ_2 is recurrently favored as the output ports in many times. And by strategically employing caching mechanisms, a limited subset comprising the most frequently utilized combinations possesses the capability to faithfully regenerate all output port bitmaps (OPBs) associated with the requested packet. It's worth noting that the power-law distribution has found application in various domains, with numerous research findings conforming to this distribution. Examples include power efficiency in data center networks (DCNs) [21], end-to-end network traffic patterns [22], and the distributional characteristics of sources within the IPv4 address space [23]. All of these studies affirm the prevalence of the 80/20 rule in network traffic dynamics. Clearly, a stronger correlation among ports leads to a higher frequency of recurring output port combinations. We demonstrate that our proposed scheme not only resolves the time inefficiency problem of the original scalar-pair vectors routing and forwarding (SVRF), but also achieves space saving compared with the original SVRF and BF schemes.

The rest of this paper is organized as follows. In Section II, the state-of-the-art of mainstream multicast forwarding algorithms and their related works, weaknesses are presented. In Section III, our proposed CSVRF scheme to alleviate memory consumption is discussed and some implementation issues are explained in detail. In Section V, we compare the performance of space and time efficiency under five metrics and implement them using hardware. Finally, some concluding remarks are presented in Section VI.

II. Related Works

A. Bloom Filter (BF)

Straightforward approach to determining multiset membership is to store the information of the group elements, usually represented as a list, in memory. To determine whether an element is a group member, we must get lists of all groups, and query each group to determine whether an element is its member. This operation features poor performance and low scalability. One of the effective ways to improve query performance is keeping an ordered full index in memory. However, this method consumes more space than the primitive methods in dealing with massive large groups [24].

The multiset group membership scheme [24]–[26] has various applications in information and communications technology (ICT) industries. A typical multiset group membership scheme consists of a compact data structure and an efficient search algorithm for determining membership about the groups to which an element belongs. More precisely, let $\mathcal{G} \subseteq \mathcal{U}$ a group and let $x \in \mathcal{U}$ be an element, where $\mathcal{U}$ is a universal set. By utilizing the group membership query we can test whether $x \in \mathcal{G}$ in both time and space. Once a group membership query is executed, a certain search algorithm will be performed to identify the target groups through a specific predefined data structure. For example, a packet forwarding algorithm needs to effectively determine whether a required packet identifier exists in a binding relationship with an egress without exhaustively checking all the elements in the forwarding table. Instead of using single-selection situations in a unicast forwarding, the group membership query is more popular for multi-answer situations provided that the query answer corresponds to multiple candidate groups, such as that in a multicast forwarding.

BF [18] is a probabilistic group membership scheme widely used in many network functions especially in forwarding/routing decision [19], [27], [28], flow identification [19], [29], and packet classification [19], [30], etc. The BF is a time-efficient algorithm with a space-efficient data structure used to represent whether an element *n* is a member of a set $\mathcal{S}$. It is constituted by an *m*-bit vector that encodes the membership of *n*-elements in a set $\{1, 2, \cdots, m\}$. The BF is initialized with all bits set to zero. It has *k*-hash functions, which hash elements uniformly and independently in the range $\{1, 2, \cdots, m\}$, where $k \ll m$. To insert an element $x \in \mathcal{S}$, the hash values are computed by *k* independent hash functions $h_1(x)$, $h_2(x)$, $\cdots$,$h_k(x)$, and the corresponding bits are set to 1s in the *m*-bit vector at random locations. The query whether $x \in \mathcal{S}$ or not is decided by the hashed values $h_1(x)$, $h_2(x)$, $\cdots$, $h_k(x)$ by checking if they are all set to 1s in this m-bit vector. If so, the query returns that $x \in \mathcal{S}$, otherwise it returns $x \notin \mathcal{S}$. In a hardware implementation, the BF performs significantly better because its *k* hash functions can be parallelized. Thus, the time complexity of BF is claimed to be roughly O(*k*) (a constant time completely independent of the number of elements contained in the set, on the premise that memory access times should be constant too).

However, the BF may mistakenly claim a nonmember to be a member due to its probabilistic property. That's to say, the BF features a small error probability of false positive while gaining considerable memory space savings. For many applications, this is acceptable as long as the false-positive rate is sufficiently small. The performance of the BF and its many variants is evaluated by three fundamental criteria: 1) Space complexity, 2) time complexity, and 3) false positive probability. The BF allows much more bits to be set while still maintaining a low false-positive probability if the parameters are perfectly chosen. Given n elements and memory space m, suppose we wish to optimize the number of hash functions *k* and bits per element *m/n*, and to minimize the false positive probability PBF of the BF, the optimization problem can be formulated as

$$\arg\min_{m,n,k\in\mathbb{N}} \mathrm{P}_{\mathrm{BF}}(m,n,k) = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k}, \quad (1)$$

where $n \leq k \leq m$. Note that

$$\mathrm{P}_{\mathrm{BF}}(m,n,k) \to (1 - e^{-\frac{kn}{m}})^{k}, \quad (2)$$

as $m \to \infty$. The optimal number of the hash functions *k* will be approximately ($m \times \ln 2$)/*n*. From that, we can determine the optimal number of functions *k* by assuming that we are given the member capacity *n* and memory requirement *m* so as to minimize the false-positive rate P_{BF} [18].

Unfortunately, the BF faces many limitations [31] as follows:

1) The BFs incur inefficient space usage thus the cost of product design may be surprisingly high.

2) The inaccuracy problems are inherent in BFs, i.e., the traffic leakage may occur once a high false-positive rate was found, it may be a security calamity for security-sensitive applications such as the military application.

3) The BFs feature easy insertion but hard deletion of elements, thus the maintenance cost of BFs grows as the number of elements increases.

4) The existing BF solutions do not have sufficient flexibility, i.e., the memory space m and the number of hash functions *k* that the BF can support must be pre-configured as fixed. Besides, the BFs only determine whether an element is in a set, but it is unable to return the value associated with an element.

B. Scalar-pair and Vectors Routing and Forwarding (SVRF)

Error-free membership query schemes, SVRF [32] and its derivative—fractional-*N* SVRF [33] were proposed to overcome the disadvantages of BF. SVRF constructs and queries group memberships based on utilizing RNS properties including the *continuous product (CP)* and *Chinese remainder theorem (CRT)* [34], [35] to improve unicast and multicast packet forwarding in a PFE. It traverses the PFE and encodes the entire forwarding table to a 'scalar-pair', which represents a single forwarding entry as a 'vector'. For example, corresponding outgoing port(s) with residues, based on the selected prime number set, i.e. 'keys'. There are three key elements in the SVRF arithmetic: 1) Node-specific scalar-pair (M_{cp}, $\mathrm{M}_{\mathrm{crt}}$); 2) flow/group-specific keys; 3) flow/group-specific vectors. Therefore, forwarding decision through a modulo operation can be done based on the properties of RNS in the PFEs.

In the initial stage of SVRF with a finite forwarding table (group) containing *n* entries (elements), all elements in the forwarding table can be rewritten in a simple form $\mathcal{K} = k_1, k_2, \cdots, k_n$, where $\mathcal{K}$ is a set containing all the unicast and multicast packet (flow) identifiers (keys), and k_x is regarded as an integer. The whole vectors in the forwarding table thus can be rewritten in the form $\mathcal{B} = b_1, b_2, \cdots, b_n$ which contains the corresponding outgoing port bitmaps (OPBs) for multicasting or outgoing port indexes (OPIs) for unicasting of all elements, and b_x denotes a bit-vectors $bx = b_{x1}, b_{x2}, b_{x3}, \cdots, b_{x\rho}$ that enumerates all possible

egress ports of the PFE, where ρ denotes the number of ports supported by the PFE. Each element bxy of the bitmap is assigned a value for all bits: A zero-bit value (bit off) indicates that the associated port is not selected in the bitmap, while a one-bit value (bit on) indicates that the associated port is selected in the bitmap, where $x \leq n$ and $y \leq \rho$. Thus we have

$$\mathcal{B} = \begin{bmatrix} b_1 = \left\{2^0 b_{11} + 2' b_{12} + 2^2 b_{13} + \cdots + 2^{p-1} b_{1,p}\right\} \\ b_2 = \left\{2^0 b_{21} + 2' b_{22} + 2^2 b_{23} + \cdots + 2^{p-1} b_{2p}\right\} \\ b_3 = \left\{2^0 b_{31} + 2' b_{32} + 2^2 b_{33} + \cdots + 2^{p-1} b_{3p}\right\} \\ \cdots \\ b_n = \left\{2^0 b_{n1} + 2' b_{n2} + 2^2 b_{n3} + \cdots + 2^{p-1} b_{np}\right\} \end{bmatrix}. \tag{3}$$

Note that all the keys are unique and relatively prime. Otherwise, k_i must be bigger than b_i, and b_i must be larger than 2^ρ in the case of multicast. In brief, a simply selected k_i might be the ith prime numbers in the range from 2^ρ+1 to $2^{\rho+1}$-1. The first scalar, $\mathrm{M_{cp}}$ is defined as a product of all n elements in $\mathcal{K}$ as follows:

$$\mathbf{M}_{\mathrm{cp}} = \prod_{i=1}^{n} k_i. \tag{4}$$

The second scalar $\mathrm{M_{crt}}$ can be constructed by the CRT function:

$$\mathbf{M}_{\mathrm{crt}} = \mathbf{CRT} \begin{pmatrix} \{k_1, b_1\}, \\ \{k_2, b_2\}, \\ \cdots \\ \{k_1, b_n, \} \end{pmatrix}. \tag{5}$$

The CRT solver function states that given primes $\{k_1, k_2, \cdots, k_n\} \in \mathbf{N}$, N represents all integers. Then for any $\{b_1, b_2, \cdots, b_n\} \in \mathbf{N}$ there exists a unique smallest positive integer solution $\mathrm{M_{CRT}} \in N$, which satisfies $\mathrm{M_{crt}}$ mod $k_r = b_r$ for any k_x and b_x. Note that the bit-vector b_x during CRT calculation is an integer value smaller than k_x. Thus, we have conducted the calculations on the scalar-pairs ($\mathrm{M_{cp}}$, $\mathrm{M_{crt}}$) of SVRF.

For each membership query, we first extract the node-specific key k_x from incoming packet identifier x as the divisor. Then we fetch the scalar-pairs ($\mathrm{M_{cp}}$, $\mathrm{M_{crt}}$) from the memory unit as the dividends. Through two long-integer division operations, the vector (OPB) b_x for element x can be obtained by

$$b_x = \begin{cases} (\mathbf{M}_{\mathrm{crt}} \mathrm{mod} k_x) & \text{,if } \left(\mathbf{M}_{\mathrm{cp}} \mathrm{mod} k_x\right) = 0 \\ 0 & \text{,otherwise} \end{cases}. \tag{6}$$

The formula (6) states that at first, the remainder of the 1^{st} division of $\mathrm{M_{cp}}$ by k_x should be checked. If this residue is zero, the other remainder obtained from 2^{nd} division, dividing $\mathrm{M_{crt}}$ by k_i will be the OPB, which helps forward packets to the designated outgoing ports. Otherwise if the remainder of $\mathrm{M_{cp}}$ (mod k_x) is nonzero, the packet should be dropped and b_x is returned zero. It can be observed that with $\mathrm{M_{crt}}$ and set $\mathcal{K}$, each element of set B can be restored by linear congruence, and in the model there is a unique solution $\mathrm{M_{crt}}$ as follows:

$$\mathbf{M}_{\mathrm{crt}} \equiv \begin{cases} b_1 (\mathrm{mod} k_1) \\ b_2 (\mathrm{mod} k_2) \\ \cdots \\ b_n (\mathrm{mod} k_n) \end{cases}. \tag{7}$$

Through this scheme, a unitary scalar-pair with sufficient keys is sufficient to forward multicast packets toward desired egress ports without any conflict.

C. Fractional-N SVRF

Based on the same concept of SVRF, the fractional-*N* SVRF [33] preprocesses a scalar-matrix by dividing an *n*-element group into *N* sub-blocks, hence when element's keys belonging to distinct sub-blocks of SVRF, it is allowed to reuse relatively smaller and identical prime keys, and membership queries can be partitioned to leverage task parallelism, resulting in less memory consumption and lower computational complexity. Nevertheless, the space efficiency is still a bottleneck hard to overcome, especially in conditions of high port-density.

To sum up, compared with the BF, SVRF can significantly reduce the memory requirement under false-free conditions. Since forwarded direction—vector can be easily computed via simple modulo operations from a scalar-pair, it provides higher flexibility in the deployment of new multicast services. SVRF is expected to achieve higher scalability and efficiency for network equipment, it features several different properties: 1) The memory usage is linearly increased until it reaches the target group capacity (limited number of elements); 2) the SVRF does not require random access across whole memory space, unless under full-load conditions. On the contrary, the SVRF sequentially accesses memory during the fetch phase of very-long integer dividend, thus it can take advantage in cost reduction by implementing the memory hierarchy, and store the scalar-pair into low-cost dynamic random-access memory (DRAM) instead of high-cost static random-access memory (SRAM), or TCAM as in BFs; 3) since a switch/router is sufficiently supported by a single SVRF constructed components, the increasing rate in memory usage is affected by both the port-density ρ (number of ports in a PFE), and membership capacity n (number of forwarding entries in a PFE). Since the primes become scarcer as they grow larger, it implies that a large n may cause exponential growth of the memory usage m.

Although SVRF overcomes the BF's major defects such as space efficiency, occurrence of false positives, and difficult member eviction, it still suffers from inefficient processing latency. The query performance of BFs might not be impacted by the high port-density, because the criterion to deploy the hardware BFs is that the number of BFs should be equal to the number of logical interfaces of a PFE [24], and the membership query is performed in parallel among all port-specific BF components. As a consequence, the total hardware cost of BF components in a high-end PFE is considerably high. In unicast SVRF indeed does not have much advantage over BF in terms of reducing the processing time, but SVRF is more prominent in multicast forwarding. Unfortunately, SVRF still faces the problem of high memory consumption when the port-density in PFEs is huge. Since a single SVRF function utilized by a PFE generates a large scalar-pair values that occupy huge memory space, resulting in serious forwarding performance degradation due to very long integer division, even if the

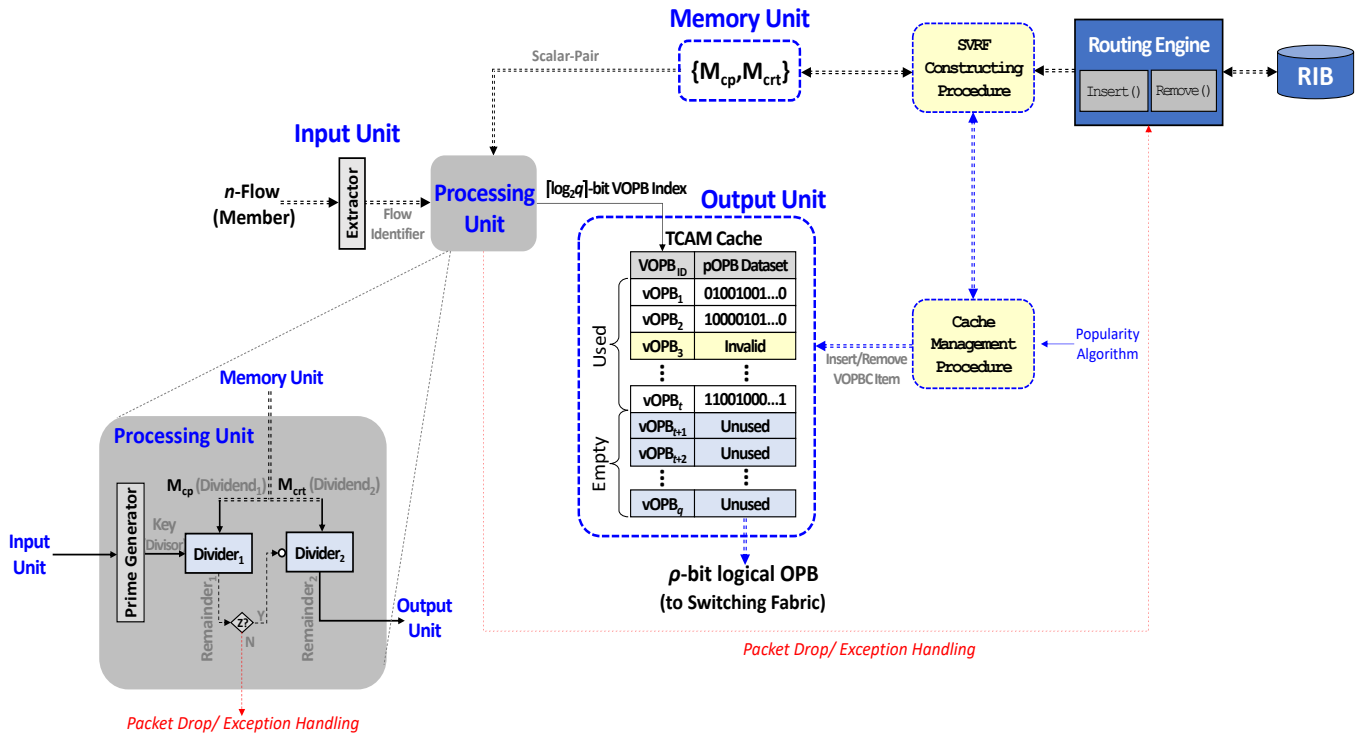

Fig. 1. The system framework of CSVRF within a ρ-port PFE.

division operation is accelerated by hardware. To underscore the efficient forwarding latency performance in fractional-N SVRF, we are driven to incorporate fractional-N SVRF into this study.

III. Caching Scalar-Pair and Vectors Routing and Forwarding (CSVRF)

In this section, we will present CSVRF through five major parts: 1) The conceptual framework of CSVRF; 2) constructing the scalar-pair; 3) querying group membership, and 4) maintaining group membership.

A. Proposed System Architecture

Though the significant time efficiency, fractional-N exhibits a limited improvement in memory space especially in high port-density. On the contrary, VOPBC boasts superior space efficiency than fractional-N regardless either high or low port-density, but conversely, it's not good as fractional-N in terms of time efficiency. A motivating idea is exploring the combination of these two schemes, aiming to achieve superior performance in both memory space and forwarding latency. Consequently, we propose a hybrid scheme named CSVRF, which integrates fractional-N and VOPBC. The goal of CSVRF is to combine the strengths of both fractional-N and VOPBC, expected to attain notable space efficiency as VOPBC while concurrently maintaining high time efficiency resembling fractional-N. Specially, in the CSVRF, firstly the whole big group membership is partitioned into N partitions, where each sub-groups operates independently. And then introduce a virtual cache which accommodates all the most frequent output port combinations. According to reuse the shorter bit-length prime and a faster searching operation via an extern virtual cache, it's reasonable to assert that CSVRF can enhance both the performance of memory space and processing time. The system model of CSVRF is illustrated in Fig. 1. The encompassing essential components of CSVRF include the 1) input unit, 2) processing unit, 3) output unit, and 4) memory unit. The characteristics of the packet processing are described as follows.

When a packet enters the system, it arrives at the input unit. This packet typically contains information such as source and destination addresses, next-hop information, and details about the payload. The input unit extracts relevant router information from the packet. This information includes details like the source address, destination address, next-hop address, and output ports. Each packet is assigned a routing identifier based on the extracted router information. This identifier is crucial for determining the appropriate processing path for the packet. The packets are then distributed among various sub-SVRF modules. This distribution is done based on the assigned routing identifiers. A demultiplexer (DEMUX) is often used for this purpose. Each sub-SVRF module operates independently. This independence ensures that each module can efficiently process the assigned packets without interference from others, each sub-group processes approximately n_N=n/N packets. Within each sub-SVRF, two dividers and a prime generator are present. Every incoming packet receives a unique prime key from the prime generator with the corresponding routing identifier, and the unique prime of each packet keep its isolated from the others. This design facilitates efficient packet

handling and routing within the system. Furthermore, each prime assigned to packets is autonomous within its respective sub-group and can be repurposed across various sub-groups for examination. This approach significantly enhances the efficiency of prime utilization, particularly in scenarios with low port-density.

Within the processing unit, individual packets undergo computation to determine the corresponding OPB. This calculation involves the modular operation between the key and scalar-pairs ($\mathrm{M_{cp}}$, $\mathrm{M_{crt}}$), which are retrieved from the memory unit. The multiplexer (MUX) consolidates the outputs from every sub-group to determine the final output within the processing unit. The output of the sub-SVRF modules is a set of virtual OPB indexs (VOPBIs), which represent the potential output ports index for the entered packets. These VOPBIs are then sent to the output unit, where they are matched with the corresponding VOPBID stored in the content-addressable memory (CAM) cache. This matching operation determines the specific entries in the cache associated with the processed packets. The recently introduced VOPBC module based on CAM is a crucial component in the output unit. The VOPBC module primarily comprises the virtual cache which stores pairs of VOPBID and the corresponding physics output port bit-map (POPB). Based on the matching results, the associated POPB dataset is transferred to logical OPB (LOPB). LOPB represents the logical OPB that corresponds to the desired output ports for the packets. The LOPB is then directed to the switching fabric. This step signifies the completion of the entire packet operation in CSVRF. The entries stored in the virtual cache are managed using a popularity algorithm. If there are modifications to the cache membership, the routing engine (RE) activates the constructing procedure. This procedure recomputes all the scalar-pairs in the memory unit to ensure the cache reflects the most popular and relevant output port combinations. Simultaneously, the cache management procedure updates the CAM cache accordingly.

CAM is acknowledged for its effectiveness in designing high-throughput forwarding engines, especially in high-speed IP lookup. renowned for its efficacy in designing high throughput forwarding engines and facilitating high-speed IP lookups. In this article, CAM is utilized for rapid matching of all entries in the virtual forwarding table with the incoming VOPBI within a single clock cycle. Despite its efficiency in quick searches, CAM does pose challenges related to high power consumption. The extensive switching of match-lines and search-lines during fully parallel search operations can lead to considerable dynamic power consumption and capacity constraints. Researchers have explored meaningful solutions to address these issues, as discussed in various research papers [36]–[38]. In the specific implementation within CSVRF, the lookup operation in the virtual forwarding table involves activating all cells in each clock cycle for parallel array access. This allows for rapid and simultaneous comparison of the incoming VOPBI with all entries in the virtual cache, resulting in the efficient determination of the LOPB for further packet processing. Because of the space constraints, the discussion on methods to reduce the power consumption of CAM will be delved as the next goal.

A notable distinction lies in the utilization of the output generated by the sub-group than conventional SVRF. Instead of directly employing it as an OPB, it is utilized as an index for the virtual forwarding table, thereby accomplishing the reusability of primes within each sub-group.

CSVRF strategically circumvents the computational complexity associated with the extensive RNS calculations present in conventional SVRF. This is achieved by employing parallel operations, focusing solely on the latency calculation for an individual sub-group and thereby minimizing computational overhead. Furthermore, though all entries in the virtual forwarding table are concurrently queried within a single clock cycle, while VOPBC introduces an additional processing cycle for match execution compared to conventional SVRF, this approach strategically optimizes processing efficiency. Nevertheless, the less bit length of each sub-group key in CSVRF leads to a significant reduction in the necessary memory space for all sub-scalar pairs ($\mathrm{M_{cp(}}_{N)}$, $\mathrm{M_{crt(}}_{N)}$). The group member query algorithm's processing time is inversely proportional to the required memory, the substantial decrease in memory demand in CSVRF translates to a corresponding reduction in processing time. Consequently, CSVRF successfully achieves a decrease in processing time through the anticipated reduction in memory space.

B. Constructing the FIB for CSVRF

Within this sub-section, we delve into the composition of the scalar-pair in CSVRF, which integrates of fractional-*N* and VOPBC, as illustrated in Fig. 2. Every entry stored in the virtual cache undergoes matching with the incoming VOPBI and leading the corresponding LOPB to the switch fabric. The added virtual cache occupies $q \times \rho$ bits; where q represents the maximum number of entries that can be accommodated in the virtual cache without overflow, and ρ denotes the number of ports in the switch. Meanwhile q must satisfy $q \geq \sum_{i=1}^{N} q_i$, the variable q_i indicates the caches number in each sub-SVRFN, and *N* is the partition number of SVRF. It's worth noting that all entries in the virtual cache are unique. The output of fractional-*N* is denoted as VOPBI and is represented by $x_{(i,N)}$ in the subsequent context. Parameter i stands for the index of ith entry in sub-SVRF$_N$. The entry can be expressed in simplified form as $\{\mathrm{K}_N, \mathrm{X}_N\}=\{\{k_{(1,N)}, x_{(1,N)}\}, \{k_{(2,N)}, x_{(2,N)}\}, \cdots, \{k_{(i,N)}, x_{(i,N)}\}\}$. In this form, K_N and X_N denotes the corresponding key set and VOPBI set of all entries in sub-SVRF$_N$. K_N and X_N construct the sub-scalar pair ($\mathrm{M_{cp(}}_{N)}$, $\mathrm{M_{crt(}}_{N)}$) of sub-SVRF$_N$ together. And the length of the key in sub-SVRF$_N$ is given by is $\lceil \log_2 q_i + 1 \rceil$, $\mathrm{M_{cp(}}_{N)}$ is the continued production of the keys, we have

$$\mathbf{M}_{\mathrm{cp}(N)} = \prod_{i=1}^{n} \left(k_{(i,N)}\right), k_{(i,N)} \in \mathbf{K}_N. \tag{8}$$

Besides, $\mathrm{M}_N = \{m_{(1,N)}, m_{(2,N)}, \cdots, m_{(i,N)}\}$, and $m_{(i,N)}$ (mod $x_{(i,N)}$) = 0, $x_{(i,N)}$ is an OPB(s) in sub-SVRF$_N$. M_N can be calculated as:

$$\mathbf{M}_N = \{\frac{\mathbf{M}_{\mathrm{cp(N)}}}{k_{(i,N)}} \mid \forall i \leq n_N \text{ and } k_{(i,N)} \in \mathrm{K}_N\}. \tag{9}$$

Another essential set is $\mathrm{C}_N = \{c_1, c_2, \cdots, c_n\}$ and can be calculated as:

$$\mathbf{C}_N = \{m_{(i,N)}\mathbf{x}(m_{(i,N)}^{-1}(\mathrm{mod}k_{(i,N)}))\}, \tag{10}$$

$\forall\, i \leq n_N$ and $m_{(i,N)} \in \mathrm{M}_N$. In (10), $m_{(i,N)}^{-1}$ is the multiplicative inverse which is calculated by $m_{(i,N)}^{-1} m_{(i,N)}(\mathrm{mod}k_{(i,N)}) = 1$. Each key within the system is generated as a unique prime number through an assumed perfect prime generator. This meticulous generation process ensures that every flow or entry in the system is distinctly and uniquely associated with its own key. After assigned the unique prime via prime generator, only a matched smallest positive integer solution $\mathrm{M}_{\mathrm{crt}(N)} \in \mathrm{Z}$ of CRT for $k_{(i,N)}$ and $x_{(i,N)}$ through

$$\mathbf{M}_{\mathrm{crt}(N)} = \mathbf{F}_{\mathrm{CRT}}\big(\big(k_{(i,N)}, x_{(i,N)}\big)\big|\forall i \in n_N\big). \tag{11}$$

Each sub-scalar pairs ($\mathrm{M}_{\mathrm{cp}(N)}$, $\mathrm{M}_{\mathrm{crt}(N)}$) is stored in the memory unit. Indeed, the crucial distinction lies in the output characteristics of fractional-N and CSVRF. In fractional-*N*, the output is an OPB with a key length of $\lceil\ \rho + 1\rceil$. In contrast, CSVRF utilizes the output of fractional-*N* as the VOPBI, serving as the index input to the virtual cache. Subsequently, the corresponding LOPB is generated and output to the switch fabric after matching, and the length of the key is $\lceil \log_2 q_i + 1\rceil$. This nuanced divergence in key properties underscores the specific design choices made in CSVRF to optimize the handling of keys and their subsequent utilization. CSVRF achieves a significant reduction in the sub-scalar pair ($\mathrm{M}_{\mathrm{cp}(N)}$, $\mathrm{M}_{\mathrm{crt}(N)}$) by optimizing the length of the key. This reduction enables effective multicast packet forwarding to multiple output ports without conflicts, thanks to the availability of sufficient keys. Despite the introduction of external VOPBC processing, CSVRF maintains a significantly smaller required memory space compared to conventional SVRF. Furthermore, it demonstrates superiority over both VOPBC and fractional-*N* in terms of memory space efficiency.

C. Performing the Forwarding Operation

When a multicast packet enters the input unit of PFE, it will get an unique key $k_{(i,N)}$ based on its routing identifier. Subsequently, this unique key becomes part of both $\mathrm{M}_{\mathrm{cp}(N)}$ and $\mathrm{M}_{\mathrm{crt}(N)}$, respectively. In the CSVRF scheme, the node-specific key $k_{(i,N)}$ serves as the divisor, and the sub-scalars ($\mathrm{M}_{\mathrm{cp}(N)}$, $\mathrm{M}_{\mathrm{crt}(N)}$) act as dividends. The residue obtained from the division of $\mathrm{M}_{\mathrm{cp}(N)}$ by $k_{(i,N)}$ is used to verify whether the queried entry belongs to the virtual forwarding table. In this case, the equitation sub-scalars ($\mathrm{M}_{\mathrm{cp}(N)}$, $\mathrm{M}_{\mathrm{crt}(N)}$) is provided in (12).

$$(\mathrm{M}_{\mathrm{cp}(N)}, \mathrm{M}_{\mathrm{crt}(N)}) = \begin{Bmatrix} (0, x_{(i,N)}) \\ (0, x_{(i,N)}) \\ \cdots \\ (0, x_{(i,N)}) \end{Bmatrix}, \text{if } \mathrm{M}_{\mathrm{cp}(N)} \mathrm{mod} k_{(i,N)} = 0. \tag{12}$$

Simultaneously, for any given $\{k_{(i,N)}, x_{(i,N)}\}$, $\mathrm{M}_{\mathrm{crt}(N)}$ can be computed based on (13). The computation of $x_{(i,N)}$ is derived from the modular operation between $\mathrm{M}_{\mathrm{crt}(N)}$ and $k_{(i,N)}$ as shown in (13)

$$x_{(i,N)} = \begin{cases} \mathbf{M}_{\mathrm{crt}(N)} \mathrm{mod} k_{(i,N)} & \text{,if } \mathbf{M}_{\mathrm{cp}(N)} \mathrm{mod} k_{(i,N)} = 0 \\ 0 & \text{, otherwise} \end{cases}, \tag{13}$$

and $\forall i \in n$. The residue of (13) represents whether the uniquely prime belongs to the set k_N. More deeply, it means the corresponding packet whether belong to the virtual cache. If the residue of $\mathrm{M}_{\mathrm{cp}(N)}$ mod $k_{(i,N)}$ is zero, indicating that $k_{(i,N)}$ belongs to set k_N, then $x_{(i,N)}$ obtained from $\mathrm{M}_{\mathrm{crt}(N)}$ mod $k_{(i,N)}$ is the index of the virtual cache. Otherwise, it denotes $k_{(i,N)}$ does not belong to set k_N , it will drops the packet or take other handling like resend.

In scenarios with the same OPB, it's supported for different multicast flows to de-map it into the corresponding OPB generated through the mentioned calculation process. After introduce popularity, a strategy employed is to cache only the most popular OPB combinations in the virtual cache. This cache is maintained periodically, ensuring the representation of various OPB combinations within the limited capacity. And it's notably that the sub-scalar-pairs ($\mathrm{M}_{\mathrm{cp}(N)}$, $\mathrm{M}_{\mathrm{crt}(N)}$) remains independent as well as the order of pair ($k_{(i,N)}$, $x_{(i,N)}$) because of the uniqueness of the prime. Further details about the CRT algorithm can be found in [35].

D. Maintaining the FIB for CSVRF

In the SVRF construction procedure, forwarding information base (FIB) was extracts from the router information base (RIB) based on the RE. The SVRF construction procedure simplifies the entire complex FIB into a structured format such as $\{\mathrm{K}_N, \mathrm{X}_N\}$, where K_N is the set of keys and X_N is the set of VOPBIs in the sub-group$_N$. This constructed information is then uploaded to the memory unit. In essence, the RE's periodic pruning based on popularity ensures that the cache remains efficient and populated with the most relevant and frequently accessed items. The item replacement strategy further optimizes cache usage by prioritizing more popular entries. The group membership maintenance function handles various operations in the MFT, such as insertion, removal, or modification of entries. When there are changes in the multicast group membership of the RIB, all multicast forwarding entries associated with the switch must be recomputed. This involves updating the values of $x_{(i,N)}$ and $k_{(i,N)}$ and subsequently updating the sub-scalar-pairs ($\mathrm{M}_{\mathrm{cp}(N)}$, $\mathrm{M}_{\mathrm{crt}(N)}$) stored in the memory unit. This process ensures that the cache accurately reflects the current state of multicast group memberships and optimizes lookup efficiency.

Upon the introduction of a new multicast flow entry to the PFE and its integration into the RIB, the RE meticulously examines the entry's multicast address, output port identifier, and next-hop node identifier. Following this verification, the entry undergoes additional processing within a sub-group, and subsequently, RE integrates it into the virtual cache. Routine maintenance of the virtual cache is a critical responsibility for RE. It systematically checks for inactive or invalid flows within the cache and removes them as necessary. The newly added

item is then seamlessly accommodated in the available space within the cache. When a new forwarding entry is introduced, RE conducts a search through the cache from OPB_1 to OPB_t. If an invalid item is encountered, it is promptly replaced with the new entry. In the absence of an invalid item, the new entry is placed in the next available slot, with the corresponding index being OPB_{t+1} when the virtual cache is not yet full. In scenarios where the cache reaches full capacity, RE employs a removal strategy based on the popularity algorithm, such as least frequency used. This strategic approach ensures that less popular items are systematically removed to create room for new entries, potentially more popular in nature. The entire operation is visually depicted in Fig. 1. Furthermore, Section IV briefly discusses a special case addressing the situation where the number of combinations exceeds the assumed capacity q.

When incorporating a new multicast entry into the virtual cache after processing within a sub-group$_N$, the introduction of a new key $k_{(i+1,N)}$ and its corresponding value $x_{(x+1,N)}$ prompts the computation of the updated scalar-pair $\mathrm{M}'_{\mathrm{cp}(N)}$. This updated pair is calculated as follows:

$$\mathrm{M}'_{\mathrm{cp}(N)} = \mathrm{M}_{\mathrm{cp}(N)} \cdot \mathrm{k}_{(i+1,N)}. \tag{14}$$

The new pair $\mathrm{M}'_{\mathrm{crt}(N)}$ is obtained by

$$\mathrm{M}'_{\mathrm{crt}(N)} = \chi_N - \mathrm{M}_{\mathrm{cp}(N)} \begin{cases} \gamma_N & ,\mathrm{if} \gamma_N \geq 0 \\ k_{(i+1,N)} + \gamma_N & ,\mathrm{otherwise} \end{cases}, \tag{15}$$

where $\chi_N = \mathrm{M}'_{\mathrm{cp}(N)} - \mathrm{M}_{\mathrm{cp}(N)} + \mathrm{M}_{\mathrm{crt}(N)}, \gamma_N = (\chi_N \mathrm{mod} k_{(i+1,N)}) - x_{(i+1,N)})$, and $\{\mathrm{K}'_N, \mathrm{X}'_N\} = \{\mathrm{K}_N + k_{(i+1,N)}, \mathrm{X}_N + x_{(i+1,N)}\}$. We find that (13) adding $\{k_{(i+1,N)} + x_{(i+1,N)}\}$ is still satisfied (11) and the original ($\mathrm{M}_{\mathrm{cp}(N)}$, $\mathrm{M}_{\mathrm{crt}(N)}$) will be updated.

If there are any entry is being removed from the virtual cache, the new scalar-pair ($\mathrm{M}'_{\mathrm{cp}(N)}$, $\mathrm{M}'_{\mathrm{crt}(N)}$) can be recalculate by

$$\left(\mathbf{M}'_{\mathrm{cp}(N)}, \mathbf{M}'_{\mathrm{crt}(N)}\right) = \left(\frac{\mathbf{M}_{\mathrm{cp}(N)}}{k_{(i,N)}}, \frac{\mathbf{M}_{\mathrm{crt}(N)}}{\mathbf{M}'_{\mathrm{cp}(N)}}\right), \tag{16}$$

where $k_{(i,N)}$ no longer belongs to set K_N and X_N, and the process of insert/ remove /modify is over.

IV. Optimization of the System Parameters for CSVRF

The integrated approach of CSVRF aims to achieve superior performance in both memory space and processing time. In this section, we will be delving into the impact on required memory space after the integration, determining the optimal number of partitions and selecting an appropriate value for q are crucial aspects in optimizing the performance of CSVRF and mitigating potential overflow problems.

A. Fractional-N with Correction τ

Absolutely, the key length plays a crucial role in the distinction between conventional fractional-*N* and CSVRF. Fractional-*N* sticks to a key length of $\lceil \rho + 1 \rceil$, aligning with the original SVRF and offering limited improvements in scenarios with high port density. On the other hand, CSVRF employs a key length of $\lceil \log_2 q_i + 1 \rceil$ for each sub-group by utilizing virtual cache. This strategic choice empowers CSVRF to surpass pure VOPBC by enabling the reusability of shorter keys and facilitating parallel operations across multiple sub-groups.

Upon integration with the VOPBC, the output of the processing unit transitions into the virtual cache as the index and is utilized to match entries stored in the virtual cache. This process enables the identification of the corresponding POPB, and the identified POPB is then output as the LOPB to the switch fabric. In comparison to the conventional fractional-*N* SVRF, CSVRF demonstrates improved memory space performance. This enhancement is attributed to the reduction in the key length. CSVRF achieves a significant reduction in key length. While introducing external memory space with the addition of a new virtual cache, the key length is notably shortened. This reduction is determined by the variable q_i ($\lceil \log_2 q_i + 1 \rceil \ll \lceil \log_2 q + 1 \rceil \ll \lceil \log \rho + 1 \rceil$). Besides, each sub-group operates as an independent SVRF module through partitioning, fostering the reusability of primes essential for each judgment. The principle of prime reusability, as elaborated in Section II, underscores the capacity of each sub-group to reuse primes independently, contributing to the efficiency of the overall system.

Another challenge lies in the fact that as the port density increases, the size of the virtual cache also grows, consuming a significant portion of the overall memory. While CSVRF mitigates memory space requirements compared to traditional VOPBC, especially at lower port densities like 16 or 64, the size of the virtual cache increases with higher port densities. This poses a challenge in terms of efficiently managing memory space as port density grows. Additionally, in terms of processing latency, CSVRF demands more cycles compared to traditional SVRF due to the incorporation of virtual caches. The processing time is closely tied to the size of scalar pairs (M_{cp}, $\mathrm{M}_{\mathrm{crt}}$). As port density increases, the processing time challenge becomes more pronounced, emphasizing the need for effective strategies to address this issue. To validate this assertion, simulations were conducted for scenarios where *N*=1 and *N*=ρ , as detailed in Section V.

B. Optimal Threshold N for CSVRF

In this subsection, we first scrutinize the change of the required memory space under the different partitions *N*. As illustrated in Fig. 2, there are two choices of *N*, one is the optimal *N* (=ρ) and another is *N* (=1) which means CSVRF delegates to conventional VOPBC. It's clear that the required memory for the conventional VOPBC (*N*=1) is much less than the CSVRF (optimal *N*). This discrepancy arises from the introduction of port correlation, allowing the dynamic adjustment of the size of q by manipulating the values of τ and φ, where φ indicates the average number of ports used for each forwarding. Indeed, as fractional-*N* preserves a consistent key length in both high and low port-density scenarios, the memory space improvement primarily stems from the reusability of primes, particularly prominent in low

port-density. The substantial memory enhancement is chiefly attributed to the virtual cache in CSVRF.

Conversely, in the conventional VOPBC scheme, n packets will enter a single SVRF processing module without any partition, all the output will directly output as LOPB to the switch fabric, and it also satisfied $q < n$. This is a case in the conventional VOPBC, when arrived entries n =4096, q=1024, the required bit-length of the key is $\lceil \log_2 q + 1 \rceil = 11$. As n and q grow, the required bit-length must grow to $\lceil \log_2 q + 2 \rceil$ and $\lceil \log_2 q + 3 \rceil$ to satisfied one-by-one mapping. Compared to the conventional VOPBC scheme, the arrived entries in each sub-group that waiting for processing is much less and only related to variable N in CSVRF ($n/N_{\mathrm{sub-group}} < n_{\mathrm{VOPBC}}$). And the cached entry is q_i in each sub-group. Finally, only required $\lceil \log_2 q_i + 1 \rceil$ bit for each sub-group, and when computing the total memory, it is only required to sum all the sub-groups. Generally, CSVRF outperforms pure VOPBC in terms of required memory space. And this improvement can be attributed to a mechanism akin to fractional-*N* such as the parallel processing and reusability of less bit-length prime.

C. Optimal Cached Entries q

In this sub-section, we explore the appropriate choice of q to prevent overflow issues. As the number of ports increases, the complexity of output port combinations also grows and requiring an increase in the corresponding preset q. It's crucial to efficiently control the size of the q to ensure that q ¡ N, and this limitation denotes a smaller forwarding table which support faster matching operation. To avoid overflow and accommodate the most popular output port combinations in the virtual cache, it is essential to adjust the value of the port correlation τ and the average output port-density φ which can be efficiently control the size of the q with the grow of port-density. Besides, a popularity algorithm is employed to maintain the popularity of entry in the virtual cache. And the value of port correlation is defined as τ=$p_i/p_j, p_i/p_j$ like 80/20, 90/10. Thus, we have

$$\begin{aligned} P_N &= \binom{n_N p_i}{\phi p_j} p_i^{n_N-\phi} p_j^{\phi} \binom{n_N p_j}{\varphi p_i}, \\ P &= \sum_{i=1}^{N} P_N. \end{aligned} \tag{17}$$

P_N is the probability of each output port combination enabled in the sub-group$_N$, and P is the sum of the output port combinations in the sub-group$_N$. Then, we have

$$q = \begin{cases} \binom{n_N p_i}{\varphi p_j}\binom{n_N p_j}{\varphi p_i}/N & \text{,if } \binom{n_N p_i}{\varphi p_j}\binom{n_N p_j}{\varphi p_i} < n \\ n, \text{ otherwise} \end{cases}, \tag{18}$$

where n_N is the arrived packets in each sub-SVRF$_N$. Absolutely, q is the core parameter both in the newly introduced virtual cache and the required bit-length of the key. And in the other hand, q denotes the correlation between ports, a higher port correlation represents a higher bias on the selection of egress port and will generate more the same port combinations with given *n*. However, there exists a special phenomenon is that the number of enabled egress ports are uncertain in each forwarding. Luckily it has been improved by introducing the parameter φ. Nevertheless, while there is no risk of overflow in high port correlation, the possibility of overflow in scenarios with low port correlation is a crucial consideration which needs to be discussed. The previously mentioned port correlation and the average port-density were applied as τ and φ, and the next port correlation and the average number of egresses is shown as τ'=p_i'/p_j' and φ'. The new $p_N'$and p' can be computed as

$$\begin{aligned} P_N{}' &= \binom{n_N p_i^{'}}{\phi' p_j^{'}} p_i^{n_N'-\phi'} p_j^{\phi'} \binom{n_N p_j^{'}}{\varphi^{'} p_i^{'}}, \\ P' &= \sum_{i=1}^{N} P_N{}', \end{aligned} \tag{19}$$

and the new q' is

$$q' = \begin{cases} \binom{n_N p_i'}{\varphi' p_j'}\binom{n_N p_j'}{\varphi' p_i'}/N & \text{, if } \binom{n_N p_i'}{\varphi' p_j'}\binom{n_N p_j'}{\varphi' p_i'}/N < n \\ n & \text{, otherwise} \end{cases}, \tag{20}$$

for $p_i > p_j'$, in this case if $\varphi \geq \varphi', dq < 0$, there is a risk of overflow. However, overflow will not occur if $\varphi < \varphi'$. Therefore, managing and adjusting the value of τ becomes crucial to avoid overflow in such cases.

V. Simulation and Results

A. Simulation Environment

In this section, simulations were conducted using Matlab2020 to assess the performance of the proposed scheme. As mentioned earlier, the processing time of a single sub-group in CSVRF is required because of all the multiple sub-SVRFs processes occur in parallel. We evaluate the performance of the proposed CSVRF scheme and compare it with fractional-*N*/traditional SVRF and pure VOPBC. The evaluation includes assessments of memory space utilization, different port correlations τ, average output port-density φ, parallel processing sub-groups *N*, processing time. Additionally, the assumption of the independence of each sub-group is considered, facilitating the reuse of the short bit-length primes in the scheme. We also discuss the size of virtual cache and scalar-pair under different port-density to help understand the trend of total memory space. Meanwhile, simulating port correlation involves creating a set of rules or relationships between different ports to mimic the behavior of a network. It can be briefly concluded as follows: First step is deciding how different ports are related to each other, then create a correlation matrix where each entry represents the correlation between ports. This matrix should reflect the rules we defined. For example, if ports A and B are strongly correlated, the matrix entry for (A, B) might be high. And consider about the randomize or vary correlations in real world all the time, we must introduce some randomness or variability in the port correlation values and selection. After generated the correlation matrix is incorporate correlation into simulation, observe the behavior of the system and analyze the impact

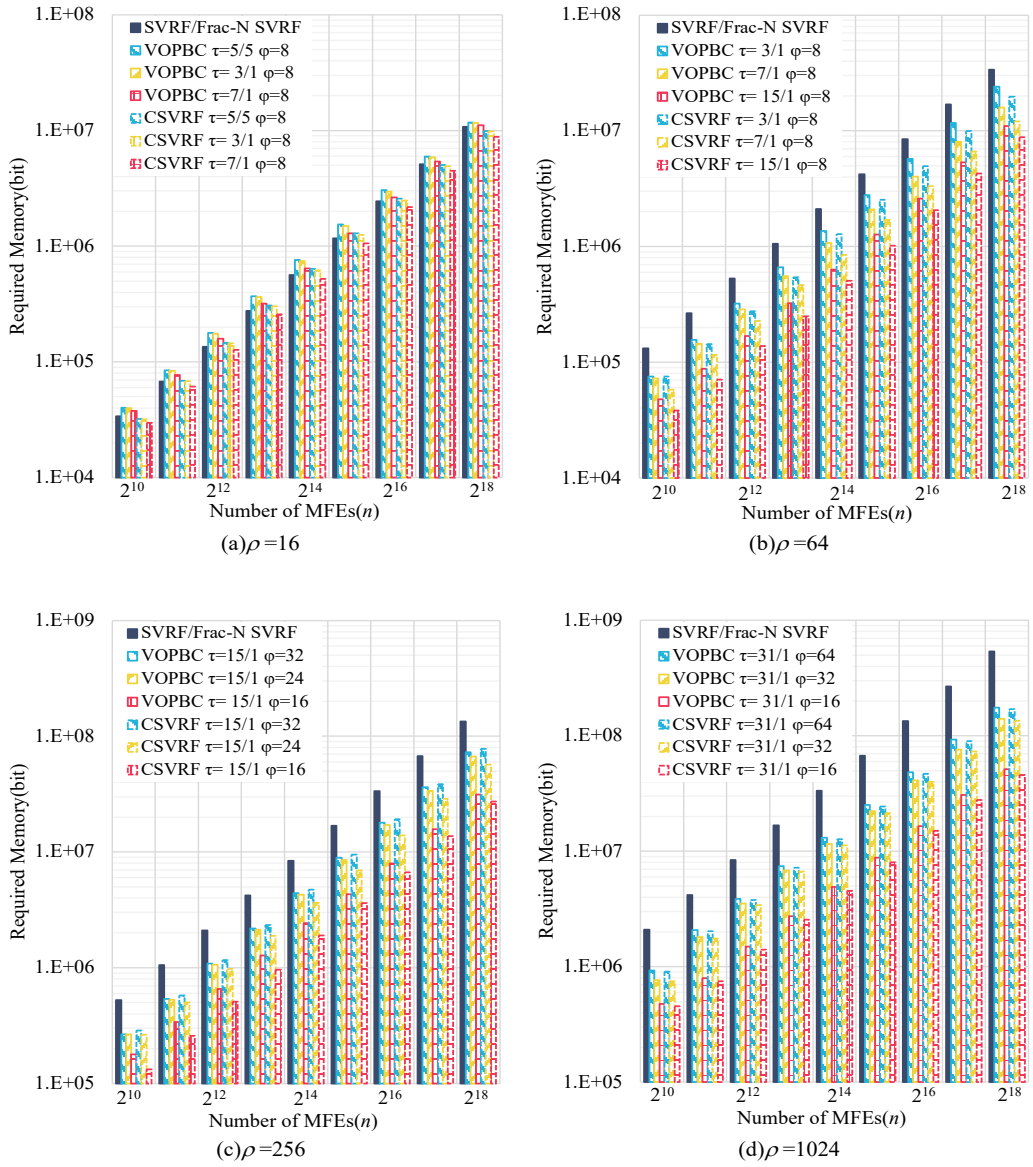

Fig. 2. Required memory space m under different forwarding entries n in the ρ-port PFEs.

of port correlation on various metrics. In our experiments, we explored four configurations of CSVRF based on the PFE with different port-density, including 16, 64, 256, and 1024-ports; the correlation coefficient τ varied from 5/5 to 31/1, the average output port-density φ is ranging from 8 to 64 respectively. Additionally, we considered two scenarios for the number of partitions: N=1 and N=ρ respectively. Our evaluation focused on two crucial metrics: space efficiency and time efficiency.

B. Memory Usages

In the evaluation of PFE, a key metric is the size of the required memory space. On the first simulation scenario, which involves assessing the size of the required memory to store the virtual cache. This measurement reflects the efficiency of the CSVRF in managing and utilizing memory resources based on different configurations and parameters, such as port-density ρ=16, 64, 256, and 1024-port, port correlation τ=5/5 (50/50), 3/1 (75/25), 7/1 (lower than 90/10), 15/1 (higher than 90/10), and 31/1(close to 96/4)), and average output port-density φ. The goal is to understand how well the PFE adapts to varying conditions and how it compares to other approaches, including CSVRF with different partition settings (N=1 and N=ρ), fractional-N, and conventional SVRF. The simulation results are depicted in Fig. 2. The description of the 80/20 rule and other correlation rules is referring to the correlation between output ports. For example, the term "50/50" suggests an equal distribution or correlation pattern. It means that half of the packets in the simulation follow one pattern of output port selection, and the other half follow a different pattern. Importantly, there is no specific correlation between the two groups; they are randomly selecting output

ports without influence from each other. 75/25 (≈3/1) rule, 7/1 (close to 90/10) etc. are subject to the same theory [39], [40]. This rule is likely used in the simulation to create a diverse and unbiased distribution of output port selection behaviors among packets, providing a basis for evaluating the performance of the PFE under different correlation scenarios.

The calculation about needed memory space in VOPBC and CSVRF as follows:

$$m_{\mathrm{VOPBC}} = \sum_{i=1}^{q} \|\mathrm{VOPI}_i^* \rho\| + \left\|(\mathrm{M_{cp}}, \mathrm{M_{crt}})_{\mathrm{VOPBC}}\right\|, \quad (21)$$

$$\mathrm{M_{cp(VOPBC)}} = \|\lceil \log_2 q \rceil + 1\| * n_E, \quad (22)$$

$$m_{\mathrm{VOPBC}} = \sum_{i=1}^{q} \|\mathrm{VOPI}_i * \rho\| + \left\|(\mathrm{M_{cp}}, \mathrm{M_{crt}})_{\mathrm{VOPBC}}\right\|, \quad (23)$$

and

$$\mathrm{M_{cp(SVRF)}} = \sum_{N=1}^{\rho} \|\lceil \log_2 q_N \rceil + 1\| * n_{E_N}, \quad (24)$$

the provided expression involves several variables as follows: Where n_E represents the total count of entries that have arrived in the PFE, $n_{E(N)}$ represents is the count of entries that have arrived specifically in single sub-group$_N$, in our system with multiple sub-groups, each sub-group may process a subset of the total entries. ρ represents the total number of ports in the router/switch, ports are connection points for devices in a network or system. $\|\log_2 q_i\|$ involves taking the base-2 logarithm of q_i and then applying the ceiling function rounds up to the nearest integer. The result of $\|\log_2 q_i\|$in this article represents the number of bits required for the key in sub-group$_N$. Figs. 3(a)–(d) shows the memory space required by CSVRF under various parameter values at different port densities. Figs. 3(a)–(b) reflects the trend of memory space with the different τ when the φ is fixed. The bigger τ will take larger improve in memory space. Conversely, the Figs. 3(c)–(d) depicts the relationship between memory space and φ with the same τ. It shows the bigger φ will occupy more memory space.

Generally, the total required memory space is linearly increasing with the port-density and our proposed scheme outperforms than conventional VOPBC and SVRF in any case, which demonstrates our scheme has a significant improvement in memory space. Nevertheless, owing to the recently introduced virtual cache, demanding additional memory, the enhancement in space efficiency is minimal when the port density is low, for instance, in the case of a 16-port configuration, as illustrated in Fig. 2(a). The rationale behind this modest improvement can be inferred through (23), where the memory space of CSVRF comprises both the external cache and the scalar-pair components. According to the result as depicted in Figs. 2(a)–(d), it's easy to see $m_{\mathrm{CSVRF}} < m_{\mathrm{VOPBC}} < m_{\mathrm{SVRF}} < m_{\mathrm{Fractional}-N}$, where *m* represents required memory space. These findings suggest that our approach demonstrates robust scalability and is well-suited for supporting large-scale networks.

C. Forwarding Latencies

Time efficiency refers to the ability of a system or process to achieve its goals or tasks within a reasonable or optimal amount of time. Lower time consumption often indicates efficiency and effectiveness in completing tasks or achieving goals. It allows for more productivity, freeing up time for other activities or endeavors. In the PFE, time efficiency is crucial for the forwarding algorithms to swiftly processing and making forwarding decisions regarding incoming packet, and certainly much lower is better. In forwarding algorithms, membership queries likely refer to the operations or queries performed to determine if a particular entry or set of information is a member of a specific subset or structure. It's a fundamental part of the decision-making process for packet forwarding and occupy the mainly processing time. To enhance time efficiency in the PFE, one typical strategy is the reduction of memory space required for membership queries. By optimizing the storage and reducing the time it takes to process and forward packets, the overall efficiency of the packet forwarding process is improved. This reduction in memory space could be achieved through introduction of fractional-*N* in this article, fractional-*N* is mentioned to enable parallel processing. Parallel processing involves executing multiple operations simultaneously, allowing for quicker completion of tasks. The combination of memory space reduction and parallel processing contributes to improved time efficiency and reduced latency.

While our proposed scheme appears to have several advantages, it's important to consider potential drawbacks or challenges. For instance, the truncation operation, especially facing the long integer division in our scheme, can indeed introduce delays and impact the accuracy of the division result. And it can be addressed through the more bit length divider. The hardware accelerator for CSVRF is designed with 2 GHz 32-bit dividers. This indicates that division operations are performed at a clock frequency of 2 GHz, and the bit-width of the dividers is 32 bits. Memory access times are crucial for overall system performance, in our scheme we assume that the demand for memory access in SRAM and DRAM on the FPGA is 10 ns and 50 ns, respectively. The data path width between memory and the processing unit is specified as 32 bits. This parameter influences the amount of data that can be transferred between memory and the processing unit in a single cycle. Each comparison/DEMUX operation are assumed to be executed in a single clock cycle.

The description outlines the findings from Figs. 3(a)–(d), which represents the relationship between the average packet forwarding latency and the number of forwarding entries. The green curves in the figures represent the forwarding latency of the CSVRF. In particular, the results in Figs. 3(a)–(d) reveal a substantial decrease in forwarding latency between CSVRF and pure VOPBC, showcasing a remarkable improvement of nearly 10× and 1000× across different port-densities. This improvement underscores the efficiency of the proposed CSVRF scheme in optimizing packet forwarding performance. Furthermore, CSVRF exhibits superior performance over the original fractional-*N*, attributed to the notable enhancement in

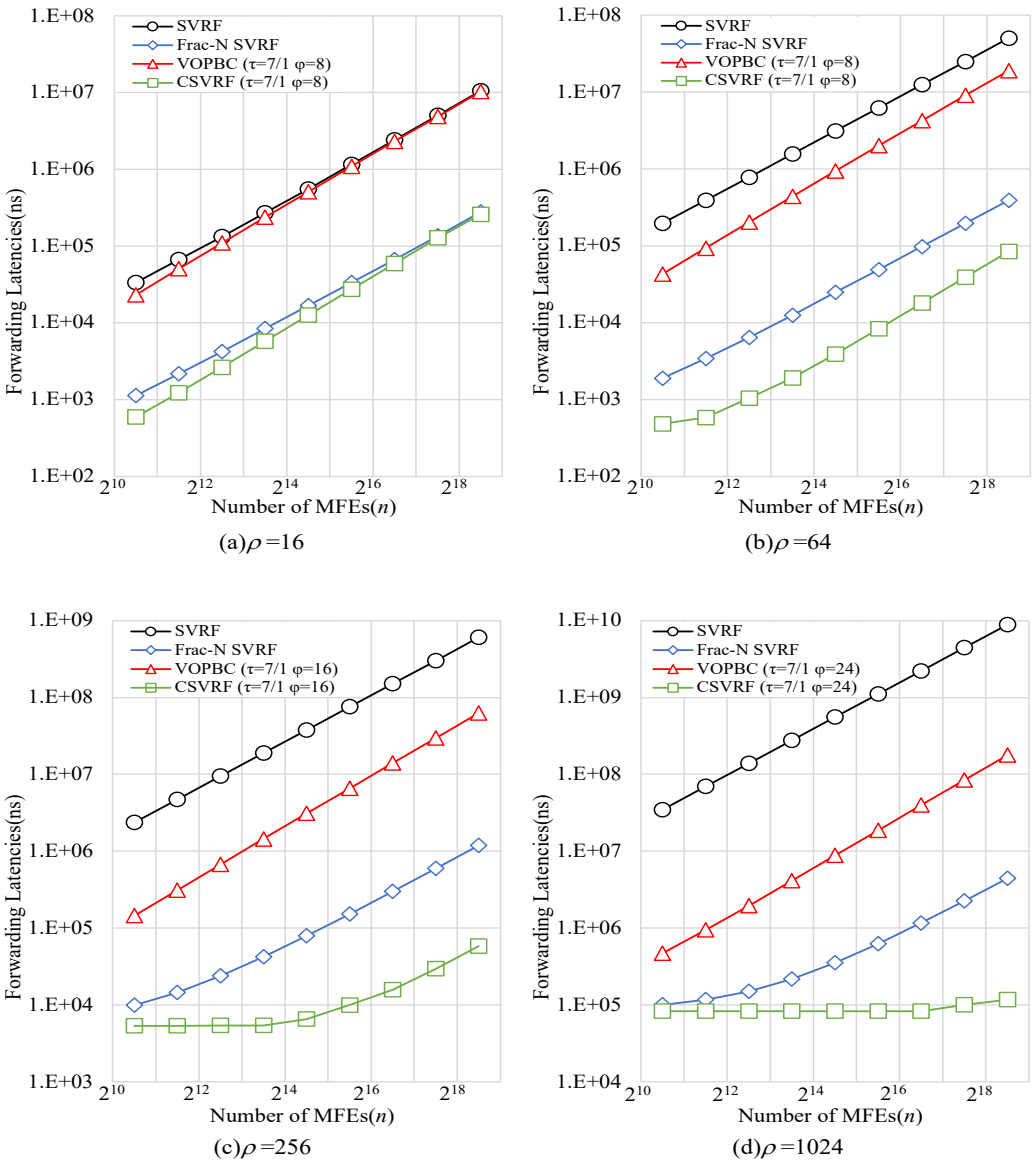

Fig. 3. Forwarding latency versus number of forwarding entries n in the ρ-port PFEs.

memory space efficiency with the increase in port-density, as demonstrated in Fig. 3. Besides, the combination of fractional-N and VOPBC, as depicted in Fig. 2(a), results in a slight improvement in memory space, which corresponds to a proportionally small enhancement in forwarding latency, aligning closely with the performance of fractional-N, as indicated in Fig. 2(a).

In summary, the integration of fractional-N and VOPBC in the CSVRF scheme proves successful in achieving better overall performance concerning both forwarding latency and memory space, surpassing the outcomes of individual schemes.

The provided description of Fig. 4 outlines the results of simulations comparing the proposed CSVRF scheme with conventional fractional-N in different scenarios involving varying values of N (the number of partitions) and τ (port correlation). As N approaches the optimal value (N=ρ), the gap between CSVRF and conventional fractional-N widens. This widening gap suggests that CSVRF demonstrates greater time efficiency compared to conventional fractional-N as the port-density increases. The decrease in memory space for each sub-group due to the increase in partitions is indeed a positive aspect as it contributes to a reduction in forwarding latency. However, it's important to note that as the number of entries increases, there will still be a corresponding increase in the overall memory space required to store the additional information associated with these entries, which can impact forwarding latencies. Even when the port correlation (τ) is set at 50/50, indicating an even distribution of traffic across the partitions or sub-groups, CSVRF exhibits better forwarding latency performance compared to conventional SVRF. And in a specific scenario where the distribution ratio is set to τ =15/1, the forwarding latency of CSVRF is highlighted to be nearly

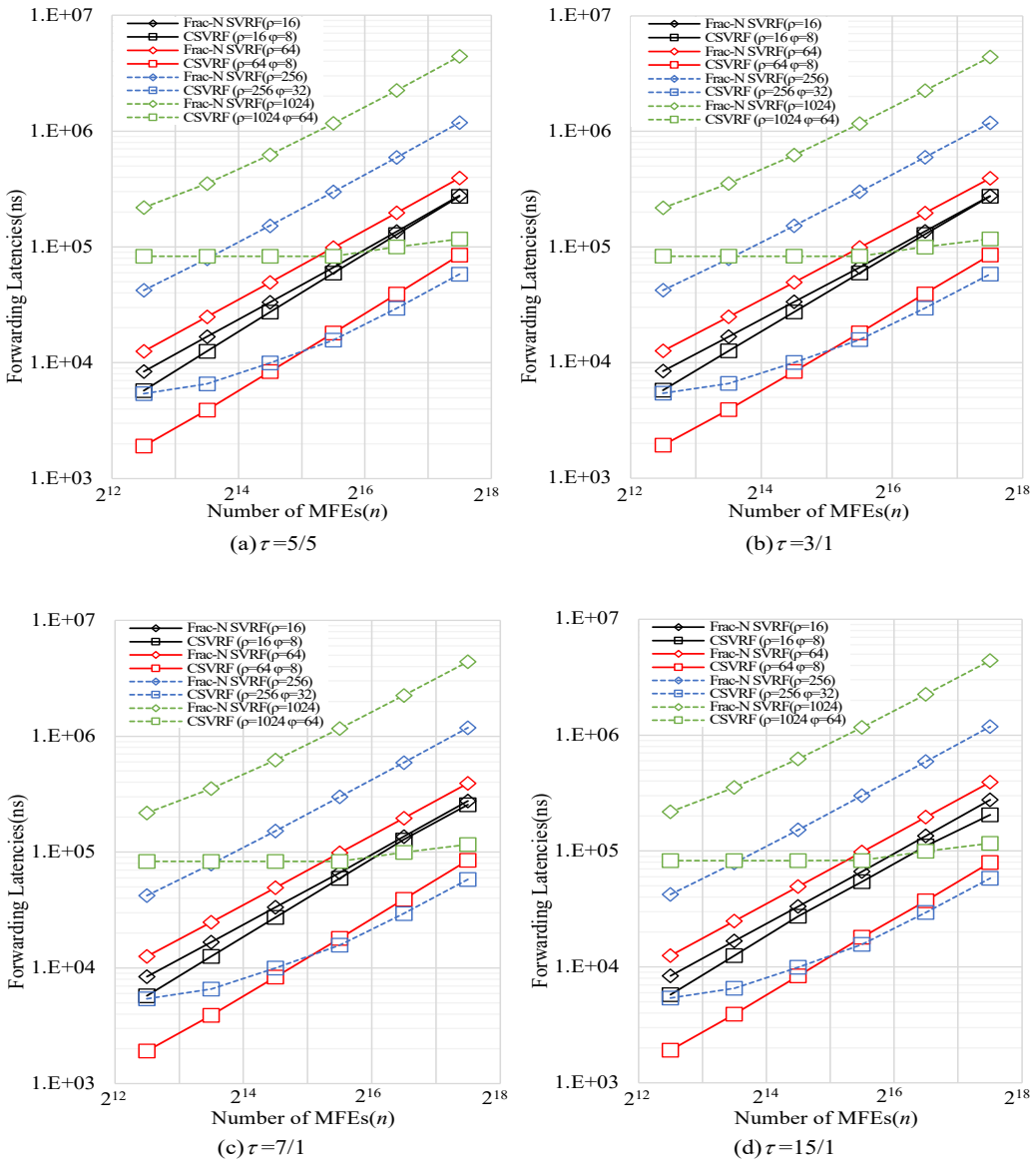

Fig. 4. Forwarding latency versus different port correlation τ and average output port-density φ in the ρ-port PFEs.

1% of the forwarding latency exhibited by conventional SVRF. In summary, while CSVRF requires more processing cycles due to the presence of an external cache, it counteracts this by efficiently optimizing the search speed through the reduction in the size of scalar-pairs. CSVRF scheme is not only better in forwarding latencies compared to conventional SVRF but also maintains its superiority even when the distribution of traffic is varied. This showcases the effectiveness of CSVRF in enhancing forwarding performance in different network conditions.

VI. Conclusions

The primary focus of designing the PFE lies in achieving rapid matching for a high volume of forwarding entries while concurrently reducing the necessary memory space. The proposed solution, termed CSVRF, introduces a hybrid strategy that amalgamates VOPBC with fractional-*N* SVRF concepts. This hybrid approach aims to overcome the constraints of conventional SVRF, enhancing performance in terms of reduced memory space requirements and improved forwarding latencies. These two metrics are crucial for assessing the efficiency and effectiveness of the proposed approach. The introduction of a virtual cache is highlighted, which includes the most popular output port combinations. Additionally, the approach involves dividing a large scalar-pair into N subgroups to achieve the reusability of primes. This reusability contributes to reducing the required memory space. CSVRF aims to decrease the required memory by utilizing a shorter length of the key. Moreover, it seeks to improve time efficiency by enabling multiple sub-module parallel computations through partitioning.

Compared to the original fractional-*N* SVRF, our modification involves adjusting the key's bit-length by introducing a virtual cache. An innovation lies in transforming the output of the original fractional-*N* SVRF to serve as the index of the virtual cache rather than directly matching with the physical OPB. In the original fractional-*N* SVRF, the key's length is represented as $\lceil \log_2\rho + 1\rceil$, where ρ is the number of ports in the switch. In CSVRF, however, the key's length becomes $\lceil \log_2 q_i + 1\rceil$, where q_i represents the number of entries to be cached in each SVRF sub-block. It's important to note that while the original fractional-*N* SVRF maintains a consistent length for the prime key, the introduction of a virtual cache in CSVRF significantly reduces the prime key's length. The transformation underscores how the virtual cache optimizes key length and boosts the performance of the CSVRF-based PFE overall.

In summary, CSVRF presents a refined approach by leveraging the benefits of both fractional-*N* SVRF and VOPBC, demonstrates superior overall performance compared to individual schemes of pure fractional-*N* SVRF and VOPBC individually, particularly in scenarios involving frequently reused output ports. Indeed, while simulation results underscore these advantages, it's crucial to recognize that the hardware implementation cost poses a substantial challenge. Striking a balance between achieving optimal hardware efficiency and preserving CSVRF's benefits represents a key area requiring further exploration and development in future implementations. Finding ways to maximize hardware efficiency without compromising the advantages CSVRF offers will be a significant focus for future endeavors. As another practical consideration, the proposed CSVRF exhibits a degree of imperfection as it is anticipated to yield heightened power consumption when implemented on PFE, attributable to its intricate hardware architecture. We will defer these aforementioned issues for future investigation and exploration.

The PFE is a critical component in high-performance switches and routers, responsible for efficient packet forwarding. The primary concern addressed in this work is the forwarding state scalability for high port-density in high-performance switches and routers. The focus is on achieving fast matching when dealing with a large number of forwarding entries and reducing the required memory space. The challenges outlined include the difficulty of achieving fast matching when millions of forwarding entries arrive simultaneously, and the performance degradation observed in conventional SVRF when port-density is high. The proposed solution is a hybrid strategy called CSVRF, which combines the concepts of VOPBC with fractional-*N* SVRF. This hybrid approach aims to address the limitations of conventional SVRF and achieve better performance in terms of memory space and forwarding latency. The performance of CSVRF is evaluated using two main metrics: Memory space and forwarding latency. These metrics are crucial for assessing the efficiency and effectiveness of the proposed approach. The introduction of a virtual cache is highlighted, which includes the most popular output port combinations. Additionally, the approach involves dividing a large scalar-pair into *N* sub-groups to achieve the reusability of primes. This reusability contributes to reducing the required memory space. CSVRF aims to decrease the required memory by utilizing a shorter length of the key. Moreover, it seeks to improve time efficiency by enabling multiple sub-module parallel computations through partitioning.

Furthermore, in comparison to the original fractional-*N* SVRF, we modify the bit-length of the required key by incorporating a virtual cache. A notable innovation lies in the transformation of the output of the original fractional-*N* SVRF, serving as the index of the virtual cache rather than directly matching with the physical OPB. In the original fractional-*N* SVRF, the length of the required key is represented as $\lceil \log\rho + 1\rceil$, where ρ is the number of ports in the switch. In CSVRF, however, the length of the required key becomes $\lceil \log_2 q_i + 1\rceil$, where q_i is the number of entries to be cached in each sub- SVRF_N. It is essential to note that the original fractional-*N* SVRF maintains a consistent length for the prime key. However, with the introduction of a virtual cache in CSVRF, the length of the prime key is significantly reduced. For a detailed proof, readers can refer to Section II in the related work about fractional-*N* SVRF. This transformation underscores the efficacy of the virtual cache in optimizing key length and enhancing the overall performance of the PFE.

Finally, summarize the main differences between the pure VOPBC and CSVRF in four aspects. 1) Handling of Scalar-Pairs: VOPBC utilizes a single, large scalar-pair for processing, while CSVRF divides the scalar-pair into *N* sub-groups, allowing for parallel computation and the reuse of shorter primes in each sub-group. 2) Key length: VOPBC requires a key length of $\lceil \log_2 q + 1\rceil$, while CSVRF achieves a shorter key length, denoted as $\lceil \log_2 q_i + 1\rceil$, due to the reuse of less-length primes in each sub-group; 3) input index: VOPBC utilizes a traditional OPB to match with all the entry in the cache, while CSVRF reusing less-length OPB in each sub-group, leading to more efficient memory space utilization. 4) Probability of overflow: VOPBC may face potential overflow issues in certain cache size scenarios, while CSVRF mitigates potential overflow concerns through the introduction of sub-groups and optimized key lengths.

In summary, CSVRF presents a refined approach by leveraging the benefits of both fractional-*N* SVRF and VOPBC, demonstrates superior overall performance compared to individual schemes of pure fractional-*N* SVRF and VOPBC individually, particularly in scenarios involving frequently reused output ports. While simulation results showcase these advantages, it's essential to acknowledge that the hardware implementation cost remains a significant challenge. Achieving optimal hardware efficiency while retaining the benefits of CSVRF presents an area for further exploration and development in future implementations.

Moreover, the proposed CSVRF scheme demonstrates a high scalability, making it well-suited for diverse networking scenarios. Scalability refers to the ability of a system to handle an increasing workload or to be easily expanded to accommodate growth without compromising performance. For instance, CSVRF can be extended to meet the demands of multicast routing and forwarding in large-scale networks. It efficiently manages the forwarding of packets,

even when dealing with numerous forwarding entries. Unlike some schemes that might be optimized for specific scenarios, CSVRF maintains superior performance even in non-80/20 scenarios. This adaptability ensures reliable operation across a range of network traffic patterns. Such as in SDN, where the control and forwarding planes are separated, CSVRF seamlessly integrates. The controller's role extends beyond prime key distribution to include the construction and maintenance of the scalar-pair. This flexibility aligns with the dynamic nature of SDN architectures. CSVRF is applicable to commercial switches such as the Tofino switch 2.0, which supports the P4 language for programming data-plane functions [41]–[43]. The programmable function in the P4 switch is designed to generate the required prime key based on the parsed information from the packet header. This prime key is crucial for subsequent operations in the CSVRF scheme serve as the basis for the modulo operation and can be precomputed by the controller. By leveraging the programmability of the P4 switch, CSVRF efficiently integrates with the data-plane processing capabilities of the switch. This approach allows for dynamic and adaptable forwarding decisions based on the specific requirements of the network and the characteristics of incoming packets. Besides, CSVRF's applicability is not limited to networking. It can address various computational applications, including high-speed parallel computing, rapid lookups, and big data analysis. This versatility enhances its scalability for use in many domains [44], [45].

REFERENCES

[1] J. Sonchack, O. Michel, A. J. Aviv, E. Keller, and J. M. Smith, "Scaling hardware accelerated network monitoring to concurrent and dynamic queries with flow," in *Proc. USENIX ATC*, 2018.

[2] D. Cerovic, V. Del Piccolo, A. Amamou, K. Haddadou, and G. Pujolle, "Fast packet processing: A survey", *IEEE Commun. Surveys Tuts*, vol. 20, no. 4, pp. 3645–3676, Jun. 2018.

[3] J. Aweya, "Introduction to switch/router architectures," in *Switch/Router Architectures: Shared-Bus and Shared-Memory Based Systems*. Hoboken, NJ, USA, 2018.

[4] J. Moy, "Multicast extensions to OSPF," IETF RFC 1584, Mar. 1994.

[5] S. Deering *et al.*, "An architecture for wide-area multicast routing," in *Proc. ACM SIGCOMM*, 1994.

[6] D. Waitzman, S. Deering, and C. Partridge, "Distance-vector multicast routing protocol," IETF RFC 1075, Nov. 1988.

[7] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT): An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM*, 1993.

[8] B. Zhang and H. T. Mouftah, "Forwarding state scalability for multicast provisioning in IP networks," *IEEE Commun. Mag.,* vol. 41, no. 6, pp. 46–51, Jun. 2003.

[9] Z.F. Liu, W.H. Dou, and Y.J. Liu, "AMBTS: A scheme of aggregated multicast based on tree splitting," in *Proc. IFIP Networking*, 2004.

[10] J. Tapolcai *et al.*, "Optimal false-positive-free bloom filter design for scalable multicast forwarding," *IEEE/ACM Trans. Netw*, vol. 23, no. 6, pp. 1832–1845, Dec. 2015.

[11] A. Benslimane, "Hierarchical Multicast Protocols with Quality of Service," in *Multimedia Multicast on the Internet*. London, UK 2007.

[12] B. Grönvall, "Scalable multicast forwarding," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 1, pp. 68–68, Jan. 2002.

[13] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "ESM: Efficient and scalable datacenter multicast routing," *IEEE/ACM Trans. Netw*, vol. 20, no. 3, pp. 944–955, Jun. 2012.

[14] K. Keykhosravi, H. Rastegarfar, N. Peyghambarian, and E. Agrell, "Overcoming the switching bottlenecks in wavelength-routing, multicast-enabled architectures," *J. Lightwave Tech.*, vol. 37, no. 16, pp. 4052–4061, Aug. 2019.

[15] J. Duan and Y. Yang, "MCL: A cost-efficient nonblocking multicast interconnection network," *IEEE Trans. Par. and Distr. Syst*, vol. 29, no. 9, pp. 2046–2058, Sep. 2018.

[16] W. Cui and C. Qian, "Scalable and load-balanced data center multicast," in *Proc. IEEE GLOBECOM*, 2015.

[17] M. Rana, M. Kaykobad, and A.B.M. Alim Al Islam, "A new approach for selecting aggregated multicast trees to reduce forwarding states," in *Proc. ACM NSysS*, 2018.

[18] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[19] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2003.

[20] F. Bonomi, M. Mitzenmacher, R. Panigrah, S. Singh, and G. Varghese, "Beyond bloom filters: from approximate membership checks to approximate state machines," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 315–326, Aug. 2006.

[21] L. Huang, Q. Jia, X. Wang, S. Yang, and B. Li, "Pcube: Improving power efficiency in data center networks," in *Proc. IEEE CLOUD*, 2011.

[22] L. Nie, D. Jiang and L. Guo, "A power laws-based reconstruction approach to end-to-end network traffic," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 898–907, 2013.

[23] P. B. Z. Chen and C. Ji, "Spatial-temporal characteristics of internet malicious sources," in *Proc. IEEE INFOCOM (Mini-Conference)*, 2008.

[24] F. Hao, M. Kodialam, T. V. Lakshman and H. Song, "Fast multiset membership testing using combinatorial bloom filters," in *Proc. IEEE INFOCOM*, Apr. 2009.

[25] M. Franceschetti and J. Bruck, "A group membership algorithm with a practical specification," *IEEE Trans. Par. and Distr. Syst.*, vol. 12, no. 11, pp. 1190–1200, Nov. 2001.

[26] J. Wei, H. Jiang, K. Zhou, and D. Feng, "Efficiently representing membership for variable large data sets," *IEEE Trans. Par. and Distr. Syst.*, vol. 25, no. 4, pp. 960–970, Apr. 2014.

[27] S. Dharmapurikar, P. Krishnamurthy, and D.E. Taylor, "Longest prefix matching using bloom filters," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 397–409, Apr. 2006.

[28] D. Thaler and M. Handley, "On the aggregatability of multicast forwarding state," in *Proc. IEEE INFOCOM*, 2000.

[29] W.C. Feng, D.D. Kandlur, D. Saha, and K.G. Shin, "Stochastic fair blue: A queue management algorithm for enforcing fairness," in *Proc. IEEE INFOCOM*, 2001.

[30] F. Baboescu and G. Varghese, "Scalable packet classification," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 2–14, Feb. 2005.

[31] A. Ousterhout, J. Perry, H. Balakrishnan, and P. Lapukhov, "Flexplane: An experimentation platform for resource management in datacenters," in *Proc. USENIX NSDI*, 2017.

[32] W.-K. Jia and L.C. Wang, "A unified unicast and multicast routing and forwarding algorithm for software-defined datacenter networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2646–2657, Dec. 2013.

[33] W.-K. Jia and Z. Jin, "Fractional-N SVRF forwarding algorithm for low port-density packet forwarding engines," *IEEE Netw. Lett.*, vol. 3, no. 2, pp. 42–46, Jun. 2021.

[34] A.S. Molahosseini, L.S. de Sousa, and C.H. Chang, "RNS applications in computer networks" in *Embedded Sys. Design with Special Arithmetic and Num. Sys.*, Cham, Switzerland: Springer, pp.369–380, Mar. 2017.

[35] C. Ding, D. Pei, and A. Salomaa, *Chinese Remainder Theorem: Applications in Comp., Coding, Cryptography. River Edge*, NJ, USA, 1996.

[36] S. W. Hussain, T. V. Mahendra, S. Mishra and A. Dandapat, "Match-line division and control to reduce power dissipation in content addressable memory," *IEEE Trans. Consum. Electron.*, vol. 64, no. 3, pp. 301–309, Aug. 2018.

[37] C. Li *et al.*, "A scalable design of multi-bit ferroelectric content addressable memory for data-centric computing," in *Proc. IEEE IEDM*, 2020.

[38] S. Xu, X. Wang, G. Yang, J. Ren, and S. Wang, "Routing optimization for cloud services in SDN-based Internet of things with TCAM capacity constraint," *J. Commun. Netw.*, vol. 22, no. 2, pp. 145–158, Apr. 2020.

[39] M. Schrage, "AI is going to change the 80/20 rule," *Harvard Bus. Rev.*, Feb. 2017. [Online]. Available: https://hbr.org/2017/02/ai-is-going-to-change-the-8020-rule.

[40] H. Zhu, "Social development paradox: An E-CARGO perspective on the formation of the Pareto 80/20 distribution," *IEEE Trans. Comput. Social Syst.*, early access.

[41] A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing SDN from OpenFlow to P4: A survey," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–37, Sep. 2023.

[42] P. Bosshart et al., "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.
[43] R. Bifulco and G. Rétvári, "A survey on the programmable data plane: Abstractions architectures and open problems," *Proc.IEEE HPSR*, 2018.
[44] L. Linguaglossa et al., "Survey of performance acceleration techniques for network function virtualization," *Proc. IEEE*, vol. 107, no. 4, pp. 746–764, Apr. 2019.
[45] P. Shantharama, A. S. Thyagaturu, and M. Reisslein, "Hardwareaccelerated platforms and infrastructures for network functions: A survey of enabling technologies and research studies," *IEEE Access*, vol. 8, pp. 132021–132085, 2020.

Ruisi Wu received the Master degree in Information and Communication Engineering from Fujian Normal University, Fuzhou, China, in 2023. He is currently pursuing the PhD degree with the Graduate School of Information Science and Technology, Osaka University. His research interests include high-performance switches and routers, named data network, and P4 switch.

Wen-Kang Jia (S'09-M'11-SM'15) received the Ph.D. degree from the Department of Computer Science, National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2011. Before returned to school, he had been a Senior Engineer and Manager since 1991 in various networking areas including ICT manufacturer, network integrator, and telecomm service provider. Since January 2018, so far he is currently a Full Professor with the College of Photonic and Electronic Engineering (P&EE) of Fujian Normal University (FJNU) at Fuzhou, China. His research interests include the OSI layer-2/3/4 such as TCP/IP protocol design, high-performance switching and routing, multicasting and broadcasting, mobile management, error resilience coding, multimedia communications, QoS and teletraffic engineering, IP-optical convergence networks, P2P overlay networks, cloud computing, and 4G/5G mobile networks. He has published more than 100 research papers, which has been cited over 500 times. He was awarded second Fujian province hundred talent plan in 2019.