A Unified Unicast and Multicast Routing and Forwarding Algorithm for Software-Defined Datacenter Networks

Wen-Kang Jia, *Member, IEEE*, and Li-Chun Wang, *Fellow, IEEE*

***Abstract*—In this article, we consider a scalability problem associated with software-defined datacenter, of which the unicast/multicast routing states is proven to be NP-complete. We introduce an efficient multiple membership query algorithm, called Scalar-pair Vectors Routing and Forwarding (SVRF), based on the prime theory such as Chinese Remainder Theorem (CRT). Our proposed algorithm simply calculates corresponding output ports of each multicast group by dividing a common scalar-pair with a group-specific key, within pseudo-polynomial time. The result is then used to make a forwarding decision within few cycles through a hardware accelerator. Compared to Bloom filter, our algorithm can achieve remarkable performance in terms of memory consumption, processing time, hardware cost, and 100% delivery accuracy, while applying for a large number of large-scale distinct flows (including unicast and multicast) in a large-scale datacenter networks. Our work may be applied to various research areas of computer science and networking.**

***Index Terms*—Multicast, Routing and Forwarding, Chinese Remainder Theorem (CRT), Bloom Filter (BF), Software Defined Networks (SDNs), Datacenter Networks.**

I. Introduction

MULTICAST is becoming increasingly important due to the huge bandwidth consumption of various network-based applications. These include multiparty conferencing, Internet collaborative works, software updates, and distributed storage replication, to name a few [1], [2]. Multicasting is a fundamental communications type, in which a data packet is sent from a source, replicated at the intermediate switches or routers, then forwarded to multiple outgoing links and eventually delivered to all destinations of the multicast group, and without unnecessarily sending that data packet to hosts hosts which do not belong to the multicast group. That means multicast traffic must be controlled in some fashion at both a network layer-2 and layer-3 .

Many datacenter administrators are beginning to accept the benefits of enabling multicasting among those hosts running multicast applications. This leads to paybacks in the aspects of bandwidth savings, better productivity and improved operating performance. The aforementioned applications require either layer-2 multicast support for discovering neighbors/services within a domain or layer-3 multicast support for operating across multiple subnets. As a result, cloud datacenter service providers are turning to the technology that creates 'multicast-enabled networks' across their datacenter in order to reducing communication costs and improving network performance as well as supporting efficient group management of clients and servers [3].

However, today's IP multicast still suffers from a scalability problem when the number of groups is large. Indeed, a multicast core router should keep all forwarding and routing information in the layer-2 and layer-3, respectively [4], for each multicast delivery tree traversing it. For a multicast edge switch, it must provide the extra functionality necessary for managing the connection status of each participate based on their membership in a certain group, during multicast sessions [5]. The disadvantage of multicast is that it excessively consumes system resources, such as memory space and processor speed. Furthermore, today's multicast also bears a notorious reputation for the deploying obstacles and many open problems such as QoS [6]. However, since system resources are limited, we must strive to maximize their utilization. In order to overcome the scalability problems in terms of number of multicast groups, quite a few multicast forwarding/routing algorithms have been proposed in the past two decades. The majority of these propositions tried to reduce the multicast states by using *Bloom filter (BF)* [7], which has indeed attracted much attention in the research community of both the computer algorithm and networking recently.

We discuss a new approach, called *Scalar-pair and Vectors Routing and Forwarding (SVRF)* in this article. SVRF uses an efficient way to construct and query group memberships based on the prime theory such as *Chinese Remainder Theorem (CRT)* [8]. Proposed scheme encodes the entire group addresses that traverse along the router/switch to a scalar-pair. A SDN control plane is introduced to inform each SDN switch regarding its scalar-pair, which is used for unicast/multicast packets delivery to determine the corresponding outgoing port(s) (a.k.a. vector) of each flow in the SDN routers/switches.

The remainder of this paper is organized as follows. In Section II, we present related works and discuss their issues. In Section III, we describe our proposed scheme and discuss some implementation issues. In Section IV, we evaluate the performance of our proposed scheme in terms of space

Manuscript received December 15, 2012; revised April 8, 2013. This work was sponsored by the National Science Council of Republic of China under Grant No. NSC 99-2221-E-009-026-MY3.

W.-K. Jia is with the Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan (e-mail: wkchia@cs.nctu.edu.tw).

L.-C. Wang is with the Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan (e-mail: lichun@cc.nctu.edu.tw). Corresponding author.

Digital Object Identifier 10.1109/JSAC.2013.131206.

efficiency, time efficiency, and hardware cost. Section V concludes the work.

II. Related Work and Problem Statement

In this section we address the major issues of multicast routing/ forwarding algorithms previously proposed which are used by the software-defined datacenter networks, and evaluate them according to several criteria, especially for scalability, as follows.

A. Deploying Multicast in the Software-Defined Datacenter Networks

A datacenter includes a set of facilities such as computer systems, network equipments and software applications. Besides, applications are composed with both compute and network resources. And for the last decade, computers and networks is a symbiotic relationship lacking any symmetry. Datacenters must now meet a number of service requirements and overcome several design obstacles in order to truly achieve business goals such as scalability. Evolutionary changes have occurred throughout various entities such as switch, router, server and storage of the datacenter. These entities are interconnected using a specific datacenter networking structure, which represents a significant portion of the total datacenter cost. Arguably, this ongoing evolution of datacenter networks has brought symmetry to the symbiotic relationship of computer and network.

Generally, datacenter networks are organized in one of two ways: layer-3 and layer-2. Traditional datacenter networks use layer-3 routing techniques (as known as IP networking) to interconnect a large amount of links that appear to IP hosts to be a single IP subnet such as Fat-Tree (FT) [9], RON [10], and PPVPN [11]. Datacenter networks based on layer-3 technologies provide many advantages, such as scalability and reliability. However, recently many critical issues arise when new features are adopted in the datacenter networks. For example, server virtualization technologies can be automatically and dynamically migrated between instances (servers), which may lead to significant management overhead in layer-3 datacenter networks. In order to support the requirements of server virtualization, network virtualization technologies can logically separate a single physical network into multiple logical networks, or consolidate multiple physical networks into a logical network, thereby simplifying the management of datacenter networks. However, traditional layer-3 network technologies are also difficult to meet all the requirements [12].

Therefore, modern datacenter network architectures typically provide layer-2 forwarding techniques (as known as Ethernet bridging) instead of layer-3 routing (e.g., DCell [13], VL2 [14], PortLand [15], BCube [16], and Flyways [17]). As a result, these approaches benefit from high throughput, high availability and lower management overhead, but suffer from lack of scalability and device portability. For example, to support virtual machine migration, a layer-2 environment is preferred but layer-2 technologies lack scalability. Thus, the first challenge is to achieve scalability with layer-3 technologies in a layer-2 environment. The other challenge of layer-2 datacenter networks is the utilization of physical links. Sufficient bandwidth may be provided in physical connection in the form of multiple links, but the spanning tree protocol which is designed to prevent forwarding loops in layer-2 allows only one link at a time. In other words, there is a single path bandwidth from one server to another. A multipath technology is required for the utilization of multiple links on the physical layer. Thus, it is difficult to design a datacenter that simultaneously achieves the objectives of scalability, VM migration, high throughput, and multipath feature.

Because current network equipments employ traditional design thinking, however, eventually it suffers from the aforementioned limitations, so it does not offer enough flexibility to satisfied that all unexpected service-level requirements for modern datacenters. For instance, in a traditional layer-3 router or layer-2 switch used by datacenter networks, the packet forwarding (data plane) and the high level routing decisions (control plane) occur on the same device, this plain structure is actually leading to inefficiency and inflexibility. For this reason, *Software Defined Networking (SDN)* is introduced as an emerging architecture and service model, which has the potential to provide scalability and flexibility to datacenter networks for reallocating network resources. SDN separates the control plane from the data plane in network switches. The data path portion (e.g., switch fabric) still resides on the switch, while high-level routing decisions are moved to a separate controller, typically a host. The OpenFlow [18] protocol is rapidly becoming the dominant SDN control protocol for an SDN controller to communicate with SDN switches. Since SDN allows a network operator to easily deploy innovative routing and forwarding protocols in a closed network, SDN is currently being implemented by major vendors, and has become an open standard that enables network operators to deploy existing and experimental protocols in the datacenter networks. The data path of an SDN switch presents a forwarding rule of flow table and actions. Each flow table entry contains a set of packet fields to match; while actions make decisions such as calculating outgoing port(s), dropping packets, and modifying fields. When an SDN switch receives a packet never seen before, and no matching flow entry can be found, then it sends this packet to the SDN controller. The controller should make a decision on how to handle this packet, and it can add a flow entry to the SDN switch so that the switch is able to know how to handle the packets which belong to the same flows in the future.

On the other side, the mainstream multicast routing protocols like MOSPF [19], PIM [20], DVMRP [21], and CBT [22], require maintenance of routing states by using a *Multicast Routing Table (MRT)*. A *Multicast Forwarding Table (MRT)* consists of a set of *Multicast Forwarding Entries (MFEs)* is directly used to control the forwarding of multicast packets in the on-tree switches/routers. The multicast algorithms require significant amount of memory space for maintaining the MFT and high process time for executing the routing and forwarding decisions. Therefore the memory in a router may fill up quickly when a large number of multicast groups appear. It is expected that the bottleneck in the future networks would be the multicast-enabled routing nodes. On the other hand, once there are many small group multicasts with highly scattered

receivers, each router may consume massive resources to keep those multicast group states; hence the expected number of groups may be limited. This problem means the traditional multicast routing protocol notorious is highly unscalable. In addition, the traditional multicast technologies are unable to provide mechanisms for either detecting end-to-end reachability and network topology or monitoring the traffic. Indeed, the source does not even know where the destinations are. As a result, reliable multicasting has never been realized.

Since a modern software-defined datacenter network is radically different from typical enterprise, campus, telecom networks, and the Internet particularly. However, existing multicast protocols built into datacenter switches/hosts are primarily based on the multicast design for the Internet. Before the wide application of multicasting in datacenter networks, it is necessary to carefully explore whether these Internet-oriented multicast technologies can well effectively accommodate datacenter networks. At the same time, facing the limit of the commercial deployment for Internet-scale multicasting, the closed and well-managed datacenters can also provide a challenging opportunity for pioneered multicast deployment. The software-defined datacenter networks provide a good opportunity for medium-scale (relative to Internet-scale) multicast deployment. However, to date there has been little research into this area. Hence, we find the technical trend of software-defined datacenter multicasting design is facing new challenges to achieve the goals.

B. Bloom filter based Multicast and its Limitations

The Bloom filters have been widely used in many network functions especially in unicast/multicast forwarding/routing decision [23]–[25], flow identification [26], packet classification [27], etc. The Bloom filter [7] is a time-efficient algorithm with space-efficient data structure used to represent whether an element n is a member of a set S. The filter is constituted by an m-bit vector that encodes the membership of n-elements in a set S. For each element $n \in S$, the bits $h_{i...k}(n)$ are set to 1s by k independent hash functions in the bit vector at random locations. To query whether an element $n \in S$, we check whether all the specific bits $h_{i...k}(n)$ in this bit vector are set to 1s. In a hardware implementation, the Bloom filter performs significantly better because its k hash functions can be parallelized. Thus the time complexity of Bloom filters is claimed to be roughly $\mathcal{O}(k)$ (a constant time completely independent of the number of elements contained inside the set). However, Bloom filter may mistakenly claim a nonmember to be a member due to its probabilistic property. That's to say, Bloom filter allows a small probability error of a 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 Bloom filter and its many variants is evaluated by three fundamental criteria: 1) space complexity, 2) time complexity, and 3) false positive probability. The Bloom filter allows many more bits to be set while still maintaining a low false positive probability if the parameters are perfectly chosen. Suppose we are given number of elements n and memory space m, and we wish to optimize the number of hash functions k and bits per entry m/n, and to minimize the false positive probability P of Bloom filter. The optimization problem can be formulated as:

$$\begin{aligned} &\underset{\{m/n,k\in\mathbb{N}\}}{\arg\min} \quad P_{fp}\left(\mathbf{BF}(m,n,k)\right) \\ &\text{, where } P_{fp}\left(\mathbf{BF}(m,n,k)\right) = \left(1-(1-\frac{1}{m})^{kn}\right)^k \quad (1) \\ &\text{, and } k < m, \end{aligned}$$

note that $P_{fp}\left(\mathbf{BF}(m,n,k)\right) \rightarrow \left(1-e^{-\frac{kn}{m}}\right)^k$ as $m \rightarrow \infty$. The optimal number of the hash functions k will approximate to $(ln\ 2 \times \frac{m}{n})$.

Unfortunately, Bloom Filter based multicast forwarding and routing engines have many limitations, including [28]:

First, the costs may be surprising high in product design. Compared with unicast, it is almost impossible to aggregate MFEs due to the multicast addresses are just logical identifiers without topological implication, thus the unicast prefix matching cannot be directly extended to multicasting. Moreover, each MFE maintains an outgoing interface set rather than a single outgoing interface, so it is a further less probable for different MFEs holding a common forwarding rule [29]. Bloom filters can be implemented in hardware but they are expensive. The criterion to deploy the hardware Bloom filters is that their amount shall be equal to the number of logical interfaces of a switch or a router [29], [30]. In order to deliver the advanced switching capabilities and density, today's L3 switches may consist of several hundreds of LAN ports, thereby the total cost of hardware Bloom filter in a single switch is considerable high. On the other hand, high memory consumption (bits per entry) can be resulted from the false positive ratio being set too low, because most of the allocated space for storing postings will be empty. Typically, Bloom filters should reside in a fast memory system such as SRAM. Therefore the cost-effectiveness of Bloom filters is further decreased.

Second, the accuracy problems are inherent in any BF applications. For the unicast routing, a multiple hit situation from the false positive has thereby confounded the forwarding engines. This situation may be tolerable, but it could incur an extra processing cost on the BF's control logics. In case of multicast routing, BF is an efficient algorithm to compress the MFEs. However, the traffic leakage may occur when the false positive has occurred. Although the multicast delivery ratios still remain the same, it may be a security calamity for security-sensitive applications such as in a military network.

Third, existing BF solutions do not have sufficient flexibility. The BFs just determine whether an element is in a set; it does not return a value associated with an element. We can imagine how useful if BFs would be return an outgoing interface list for multicast routing decision. One scheme that uses BFs to return a (small) set of values is outlined in [31]. However, this technique does not allow state changes. Therefore, the usage flexibility of BF is constrained.

Finally, the maintenance of BF is also costly. The BFs allow easy insertion but not deletion. Deletions in a BF is handled by using a counting Bloom filter, which keeps a counter

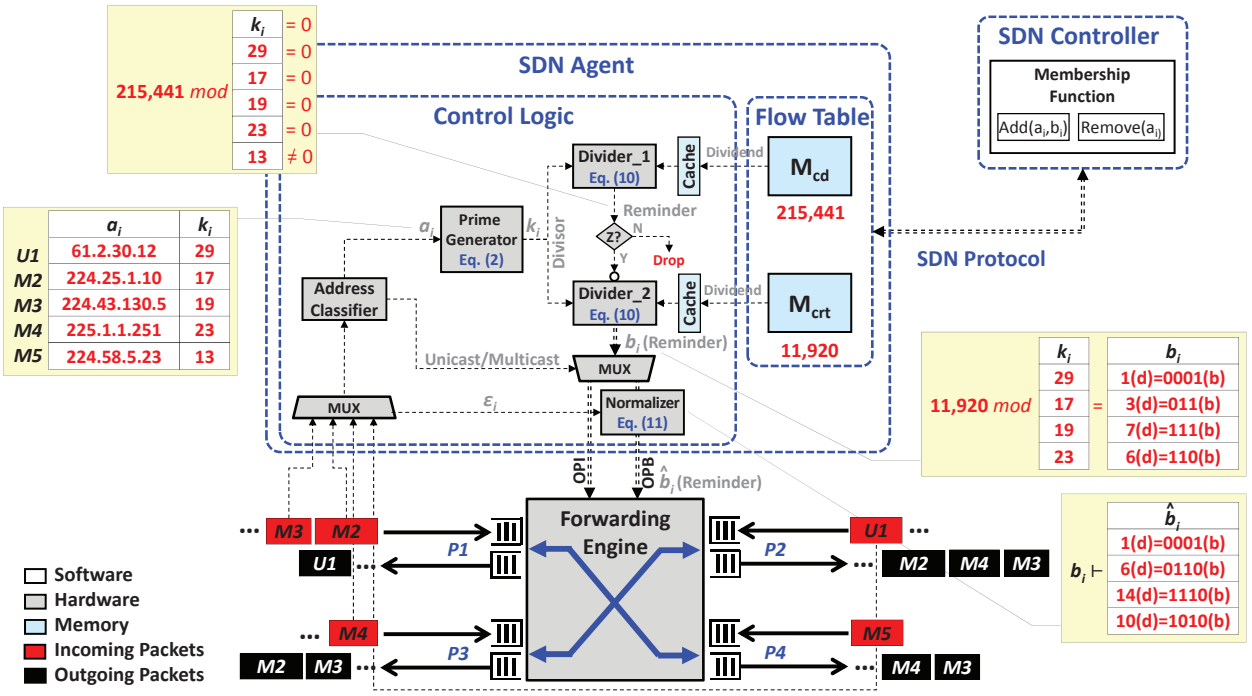


Fig. 1. Block diagram of SVRF operating in a SDN switch.

instead of a single bit at each hash location [32]. Counters are incremented in an insertion and decremented in a deletion. Unfortunately, using counters increases the memory size of the BF; it implies that the hardware cost of counting Bloom filter will be also increased.

Bloom filter has experienced various enhancements such as Counting Bloom filters (CBF) [32], Compressed Bloom filters [33], Generalized bloom filters (GBF) [34], Group-Hierarchical Bloom filter array [35], Space-Code Bloom filters [36], Spectral Bloom filters (SBF) [37], Dynamic Bloom filters (DBF) [38], Multi-Dimension Dynamic Bloom filters [39], Distance-Sensitive bloom filters (DSBF) [40], Parallel Bloom filters (PBF) [41], Load Balanced Bloom filters [42], Combinatorial Bloom filters [43], Incremental Bloom filters (IBF) [44], Variable-Increment Counting Bloom filters (VICBF) [45], and Bloomier filter [31]. Other details can be referred to the survey of Bloom filters [29]. The emerging Bloom Filter technologies and their services are nowadays still in development.

III. System Framework

We propose a novel multicast forwarding/routing algorithm to deal with various existing issues regarding multicast packet forwarding in today's software-defined datacenter networks. Throughout the proposed scheme, the considered datacenter networks are characterized entirely by SDN controller and switches, and the scheme will be useless for those networks with any non-SDN switches. Our goals are to improve the scalability and efficiency of multicast forwarding, and provide higher flexibility in the deployment of new multicast services in future software defined networks.

A. Preliminaries

There are three key elements for the proposed arithmetic: 1) node-specific scalar-pair $(\mathbf{M}_{cp}, \mathbf{M}_{crt})$, which are stored at each switch's internal memory such as flow table; 2) node-assigned group(flow)-specific key (a.k.a. identifier), which are transformed from each destination IP address; 3) node-specific vectors (a.k.a. *Output Port Bitmap (OPB)* or *Output Port Index (OPI)*) at each switch. Based on the properties of the prime number theorem such as CRT, a scalar-pair divided by different keys will remain as different vectors (OPBs/OPIs) as a remainder. We use this property to represent the exact membership query mechanism in the SDN forwarding engines.

As shown in Fig.1, an SDN controller instructs the SDN switch regarding what actions they should take using the SDN protocol such as Openflow. The switch supports both SDN-enabled flow forwarding and the proposed scheme. Initially, SDN controller discovers the network topology using the *Link Layer Discovery Protocol (LLDP)* [46]. If a unicast path or a multicast delivery tree is originated from the SDN controller, two scalars will be constructed and delivered to the flow table of the SDN switch. As the name implies, the scalar-pair contains entire forwarding information (a.k.a. vectors) for all unicast and multicast flows which traversed the switch. When a packet arrives at the SDN switch, the destination address fields are extracted and mapped (or hashed) to a unique key, which is then processed by the proposed arithmetic (or by implementing a hardware accelerator). Then the packet is either forwarded to the designated outgoing port(s) or dropped, depending on the result of the proposed arithmetic with the given input key and scalar-pair. Details of these operations are described in the following subsections.

B. Constructing Scalar-pair

Before sending the packets over the SDN data plane, the underlying routing protocol will determine optimal ene-to-end routing information along the unicast path or multicast tree by SDN controller. A forwarding entry can be represented by

the pair (a_i, b_i) that indexes the i-th entry, where a_i denotes the routing identifier such as a destination IP address or a (S,G) entry [47], and b_i denotes its corresponding output port bitmap(s) (a.k.a. vector(s)). We can rewrite the whole forwarding table entries in simplify form $\mathrm{A} = \{a_1, a_2, \ldots, a_n\}$ and $\mathrm{B} = \{b_1, b_2, \ldots, b_n\}$ respectively, where A can be seemed as a MFT which contains all the unicast and multicast flow identifier. We consider that a finite set A contains n elements. Let $\mathrm{K} = \{k_1, k_2, \ldots, k_n\}$ denotes a new key set comprising corresponding a_i-th prime numbers. A transfer function is defined as

$$\mathrm{K} = \big\{\mathcal{P}(a_i) \;\big|\; \forall i \leq n, \text{ and } a_i \in \mathrm{A}\big\}, \tag{2}$$

where $\mathcal{P}(a_i)$ denotes an assumed perfect prime-generating function that will return an unique prime number in the designated range of primes, where $a_i \in \mathrm{A}$ and $k_i \in \mathrm{K} \subset \mathbb{P}$, which k_i must be larger than a given integer 2^ρ (OPB/OPI size), and pair-wise co-prime with each other k_j, and where ρ denote the number of ports of switch. Thus the unique prime number k_i associated with each element a_i provides a single unified numbering system for proposed scheme, regardless of their order.

To ensure that each group (flow) obtains a unique key, and all the keys in the switch are pairwise relative primes, either 1) sequentially (continuously) or 2) randomly manners might be adopted: In sequential approach, when the multicast group is configured, each multicast group can be pre-assigned a unique key value from the shorter prime number in numeric order by using a predefined primes library that satisfies the requirements as previously described. Since the keys are the smaller the better in the proposed arithmetic, the table-lookup manner trades time for space in our proposed arithmetic. However, it is costly both in extra space of table, thus using hashing may be a better choice. In random approach, each multicast group obtains an arbitrarily prime number such as the a_i-th prime number by using a hash function. Hashing does not need a table to mirror the mapping information between a_i and k_i. We assume that a perfect hash function maps distinct addresses to a set of keys (primes) with no collisions. Note that for each multicast group (flow) in specific SDN switch, the group-specific key differs from each other. For each switch, the node-specific scalar-pair is shared for different unicast and multicast flows, and a specific OPB/OPI for each flow is extracted from its corresponding key.

There are two types of b_i in the system, if $a_i \in \mathbb{U}$, the b_i denotes an OPI. if $a_i \in \mathbb{M}$, the b_i denotes an OPB, where $\mathbb{U}$ and $\mathbb{M}$ denote unicast and multicast address pool respectively. From the multicast's perspective, it is undesirable a multicast packet to be copied back to the incoming port of an on-tree switch. In addition, smaller OPBs are better in the proposed scheme. Thus, an artificial OPB b_i is transformed by real OPB $\widehat{b_i}$, which is obtained from a normalization operation by

$$b_i \dashv \widehat{b_i} = \left(\frac{\widehat{b_i} \wedge \overline{(2^{\varepsilon_i+1} - 1)}}{2}\right) + \left(\widehat{b_i} \wedge (2^{\varepsilon_i} - 1)\right), \tag{3}$$

where ε_i denotes the incoming port index on switch of multicast group i, which will exclude of b_i, so we ensure that $|b_i| \leq \left|\widehat{b_i}\right|$.

Then, a new scalar $\mathbf{M}_{cp}$ is defined as a 'continued product' of all n elements in the set K:

$$\mathbf{M}_{cp} = \prod_{i=1}^{n} (k_i), \text{ where } k_i \in \mathrm{K}, \tag{4}$$

Using $\mathbf{M}_{cp}$ and set K, we can create a new set $\mathrm{M} = \{m_1, m_2, \ldots, m_n\}$ so that $m_i (\mathrm{mod}\ b_i)$=0 for all m_i, which is satisfied in the following equation

$$\mathrm{M} = \Big\{\frac{\mathbf{M}_{cp}}{k_1} \;\Big|\; \forall i \leq n, \text{ and } k_i \in \mathrm{K}\Big\}, \tag{5}$$

and another new set $\mathrm{C} = \{c_1, c_2, \ldots, c_n\}$ is created by

$$\mathrm{C} = \big\{m_i \times \big(m_i^{-1} (\mathrm{mod}\ k_i)\big) \;\big|\; \forall i \leq n, \text{ and } m_i \in \mathrm{M}\big\}, \tag{6}$$

The term m_i^{-1} denotes the multiplicative inverse of $m_i \bmod k_i$ defined by $m_i^{-1} m_i (\mathrm{mod}\, k_i)$=1. This requires that m_i and n_i should have no common divisors larger than 1, i.e. $\gcd(m_i, k_i) = 1$. Because m_i is the product of all n-elements minus k_i, all the n-elements should be relative primes to satisfy the requirement. The second scalar $\mathbf{M}_{crt}$ can be easily calculated using sets B and C

$$\mathbf{M}_{crt} = \left(\sum_{i=1}^{n} (b_i \times c_i)\right) (\mathrm{mod}\ \mathbf{M}_{cp}). \tag{7}$$

The value of $\mathbf{M}_{crt}$ is the minimum positive integer which satisfies the aforementioned definition, then it can be easily shown that maximum negative integer solution $\neg\mathbf{M}_{crt}$ obtained by $\mathbf{M}_{crt}$-$\mathbf{M}_{cp}$. In practice, the maximum negative integer solution may reduce the store size of $\mathbf{M}_{crt}$ slightly.

Now, we have the scalar-pair ($\mathbf{M}_{cp}$, $\mathbf{M}_{crt}$). Based on the properties of the prime theorem, a $\mathbf{M}_{cd}$ divided by different keys will remain as zero or nonzero, which can be used to answer inquiries regarding whether one element k_i (associated to a_i) belongs to a set K (associated to set A). Based on the properties of the CRT, a $\mathbf{M}_{crt}$ divided by different keys will remain as different OPBs/OPIs as a remainder, that's to say

$$\mathbf{M}_{cp} \equiv \begin{Bmatrix} 0\ (\mathrm{mod}\ k_1) \\ 0\ (\mathrm{mod}\ k_2) \\ \cdots \\ 0\ (\mathrm{mod}\ k_n) \end{Bmatrix}, \tag{8}$$

and

$$\mathbf{M}_{crt} \equiv \begin{Bmatrix} b_1\ (\mathrm{mod}\ k_1) \\ b_2\ (\mathrm{mod}\ k_2) \\ \cdots \\ b_n\ (\mathrm{mod}\ k_n) \end{Bmatrix}. \tag{9}$$

We use this property to represent the entire unicast and multicast flows and their vectors (b_i) for a switch. In SDN architecture, the scalar-pair is calculated by the controller, it then sends the scalar-pair to the switch to be added to flow tables, via SDN control protocol.

C. Querying Group Membership

Each incoming packet is in-need duplicated and forwarded to egress interface(s) indicated by the remainder b_i at each

switch i, which is constructed by using a simple modulo operation: Let the scalars ($\mathbf{M}_{cp}$, $\mathbf{M}_{crt}$) be the dividends and node-specific key k_i be the divisor. Through two long integer dividers, the desired OPB/OPI b_i for each multicast group i is obtained by

$$b_i = \begin{cases} \mathbf{M}_{crt} (\mathrm{mod}\ k_i) & ,\text{if } \mathbf{M}_{cp} (\mathrm{mod}\ k_i) = 0 \\ 0 & ,\text{if } \mathbf{M}_{cp} (\mathrm{mod}\ k_i) \neq 0 \end{cases}. \quad (10)$$

If the first remainder which obtained through dividing $\mathbf{M}_{cp}$ by k_i that is zero, the second remainder b_i is obtained by dividing $\mathbf{M}_{crt}$ from k_i. Else if the remainder of $\mathbf{M}_{cp} (\mathrm{mod}\ k_i)$ that is nonzero, and $b_i = 0$ means the packet is being dropped. Note that the procedure does not distinguish between unicast and multicast.

Before packets get forwarded towards their next-hops, note that $b_i \in \mathbb{M}$ has been normalized by disregarding incoming port ε_i, and each switch restores the correct $\widehat{b_i}$ by using

$$\begin{aligned} b_i \vdash \widehat{b_i} = &((b_i + 1) \wedge (2^{\varepsilon_i} - 1)) \\ &+ 2 \times [(b_i + 1) \oplus ((b_i + 1) \wedge (2^{\varepsilon_i} - 1))], \end{aligned} \quad (11)$$

through OPB $\widehat{b_i}$ is converted to a Boolean, and forwarding this packet to designated outgoing port(s). Hence, it can be observed that by having $\mathbf{M}_{cp}$ and the set K, each element of the set B can be restored by linear congruence Eq.(9) and a unique solution can be derived on the model.

That is to say, when the packets traverse a specific switch and router, the different destination addresses from the IP headers will be transferred to the unique keys, and so different outgoing port index and bitmap will be obtained from resolving unitary ($\mathbf{M}_{cp}$, $\mathbf{M}_{crt}$) values with different keys. Therefore, a unitary scalar-pair with multiple different keys is sufficient to distinguish different OPB from another with no conflict, and theoretically there can hold more multicast forwarding entries in the limited memory space. In addition, the order of pair (k_i, b_i) is irrelevant in the data structure. A detailed proof of the CRT algorithm can be found in [8].

D. Maintenancing Group Membership

From the perspective of multicast switches, a grafting operation may be triggered by the joining operation of a new member (leaf) from outside the current multicast distribution tree, while the multicast session is in progress. In contrast, pruning is the process to terminate connections that are no longer active in a multicast tree. A pruning operation may be triggered by all the participating member(s) that leaved the multicast distribution tree, thus data packets of multicast session are no longer replicated for that sub-tree (interface).

When a multicast group's membership changes, all MFEs associated with the particular SDN switches must be recalculated. Group membership maintenance function will be used so as to insert/delete an entry to/from the MFT, or modify an existing entry inside the MFT. The function is provided to the forwarding engines by an interface that consists of the following basic operations: 1) $\mathrm{Add}(\mathrm{A}, \mathrm{a_i}, \mathrm{b_i})$ is a process to add an element a_i to a group A. 2) $\mathrm{Remove}(\mathrm{A}, \mathrm{a_i})$ is a process to remove the member a_i from the membership of group A. 3) $\mathrm{Modify}(\mathrm{A}, \mathrm{ai}, \mathrm{bi})$ is a combined process to change a participating member a_i with new corresponding OPB/OPI b_i from the membership of A.

Once a new multicast group a_i joins to the switch, the $\mathrm{Add}(\mathrm{A}, \mathrm{a_i}, \mathrm{b_i})$ operation should be performed by the SDN controller. Given the new key k_{n+1} and the corresponding OPB and b_{n+1} of a_{n+1} respectively, we recalculates the new $\mathbf{M'}_{cp}$ by

$$\mathbf{M'}_{cp} = \mathbf{M}_{cp} \times k_{n+1}. \quad (12)$$

Then the new $\mathbf{M'}_{crt}$ can be obtained by

$$\begin{aligned} \mathbf{M'}_{crt} = \chi - &\begin{cases} \mathbf{M}_{cp}(\gamma - b_{n+1}) & ,\text{if } \gamma \geq b_{n+1} \\ \mathbf{M}_{cp}(k_{n+1} + \gamma - b_{n+1}) & ,\text{otherwise} \end{cases}, \\ &,\text{where } \chi = \mathbf{M'}_{cp} - \mathbf{M}_{cp} + \mathbf{M}_{crt} \\ &,\text{and } \gamma = \chi\,(\mathrm{mod}\ k_{n+1}) \end{aligned} \quad (13)$$

thus the new set K′= K + $\{k_{n+1}\}$, and new set B′= B + $\{b_{n+1}\}$, which represent the new set A′=$\{a_1, a_2, \ldots, a_n, a_{n+1}\}$. We find that Eq.(10) adding (k_{n+1}, b_{n+1}) is still satisfied Eq.(9). These operations implied in original $\mathbf{M}_{crt}$ and $\mathbf{M}_{crt}$ are replaced by newer ones.

In contrast, once all the outgoing interfaces of group a_i in the switch are pruned, the $\mathrm{Remove}(\mathrm{A}, \mathrm{a_i})$ will be performed, the SDN controller will recalculates new $\mathbf{M'}_{cp}$ by given prime key k_i. Simple calculations show

$$\mathbf{M'}_{cp} = \frac{\mathbf{M}_{cp}}{k_i}, \quad (14)$$

the quotient is the new $\mathbf{M'}_{cp}$ which replaced the original $\mathbf{M}_{cp}$, then we have

$$\mathbf{M'}_{crt} = \frac{\mathbf{M}_{crt}}{\mathbf{M'}_{cp}}. \quad (15)$$

Thus that k_i no longer belongs to K and $\mathbf{M'}_{crt} (\mathrm{mod}\ k_i)$, it implies that $a_i \notin$ A. Without loss of generality, our proposed fast $\mathrm{Remove}(\mathrm{A}, \mathrm{a_i})$ arithmetic provides much lower complexity compared to these equations provided by previous subsection.

E. Operational Examples

In this subsection, we give examples demonstrating that the unicast/multicast packets forwarding by proposed scheme through 4-port SDN switches is described in Fig.1. Before packets (flows) $\{U1,\ M1,\ M2,\ M3\}$ traverse the switch, the functions of routing of SDN controller are performed from the underlying unicast/multicast protocol. When multiple unicast path and multicast delivery tree are established using its associated addresses, a MFT contains entire flows can be represent in a scalar-pair ($\mathbf{M}_{cp}$, $\mathbf{M}_{crt}$), that has been generated by the SDN controller and then delivered to the flow table of the SDN switch beforehand.

In the case of unicast forwarding, when a unicast packet $U1$ arrive at the inbound interface $P2$ of the switch, the forwarding engine firstly extracts the destination address a_1 of the incoming packet and retrieves the corresponding key k_1=29 from the proposed Eq.(2). Based on the proposed arithmetic as described in previous subsection, the forwarding engine obtains the output port index b_1=1(d)=0001(b) by performing the $215,441 (mod\, 13) = 0$ and $119,20 (mod\, 29)$ operation (detailed as in Eq.(10), then through the internal

forwarding fabric of the switch, the $U1$ is duplicated and finally to the proper egress interface $P1$.

In the case of multicast forwarding. When a multicast packet $M2$ arrive at the inbound interface $P1$ of the switch, the forwarding engine firstly extracts the destination address a_2 of the incoming packet and retrieves the corresponding key k_2=17. Then the forwarding engine obtains b_2=3(d)=011(b) by performing the $215,441(mod\,13) = 0$ and $119,20(mod\,17)$ operation, through the normalization (detailed as in Eq.(11)), we obtain the correct output port bitmap $\widehat{b_2}$=6(d)=0110(b), and the $M2$ is finally duplicated and dispatched to the proper egress interfaces $\{P2, P3\}$.

Then the switch receives the multicast packet $M4$ through $P3$, obtains the k_4=23 and b_4=6(d)=110(b) by performing the $119,20(mod\,23)$ operation, and forwards the packet to next-hop switches via egress interfaces $\{P2, P4\}$ according to ${}^b4 = 10(d) = 1010(b)$ after normalization.

Then followed a multicast packet $M3$ which arrived at the router from the $P1$, the forwarding engine obtains the k_3=19 and b_3=7(d)=111(b) by performing the $119,20(mod\,19)$ operation. Through normalization, we have ${}^b3 = 14(d) = 1110(b)$, and forwards the packet to next-hop switches via port $P2$, $P3$, and $P4$ respectively. Thus this packet has arrived at all desired egress interfaces.

Once a packet $M5$ arrives that obtains the k_5=13 but does not match any MFT entry by performing the $215,441(mod\,13) \neq 0$ operation. The switch may have been configured to simply drop packets for which no flow has been defined, else sent this packet $M5$ to the controller, to instruct it to create a new flow.

IV. Comparative Analysis

In order to evaluate the performance of the proposed scheme, we compare its performance with the general Bloom filter under various conditions. There are three metrics that should be considered for this evaluation, including 1) space efficiency; 2) time efficiency; and 3) hardware cost. As far as accuracy is concerned, the proposed scheme and Bloom filter are incommensurable, because the membership query results of our proposed scheme are always exact.

A. Space Efficiency

Unlike Bloom filter, the proposed scheme features several different properties: 1) The memory usage is progressively increases until it reaches the target group capacity (limited number of elements); 2) Based on the aforementioned principles, the SVRF does not require random memory access, and has strictly sequential memory access during the very long integer division process. Thus we can take advantage of cost down by implementing the memory hierarchy strategy, and we can store the scalar-pair into low-cost DRAM and cache instead of the high-cost SRAM, CAM [48], [49] or TCAM [50] like that in BFs; 3) Since a switch/router is sufficiently supported by a single SVRF construction, where the increasing rate of memory usage is affected by both the port capacity ρ (maximum number of interfaces in system) and target membership capacity ϕ (maximum number of flows in system), because that the primes become more scarce as they grow larger. Hence, ρ and ϕ that the proposed scheme can support must be pre-configured.

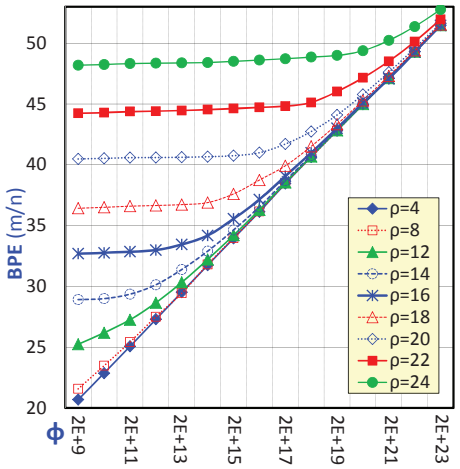


Fig. 2. Required BPEs ($\frac{m}{n}$) vs. membership capacity ϕ in ρ-port switches.

Regarding the memory space usage m in proposed scheme, it results in

$$m = |\mathbf{M}_{cp}| + |\mathbf{M}_{crt}| \quad . \tag{16}$$

The average bit length of $\mathbf{M}_{cp}$ can be derived as

$$|\mathbf{M}_{cp}| \approx \left(\sum_{\rho<i\leq\varphi+1} \left[\left(\pi(2^i) - \pi(2^{-1})\right) \times i \right] \right) \geq |\mathbf{M}_{crt}|\,, \tag{17}$$

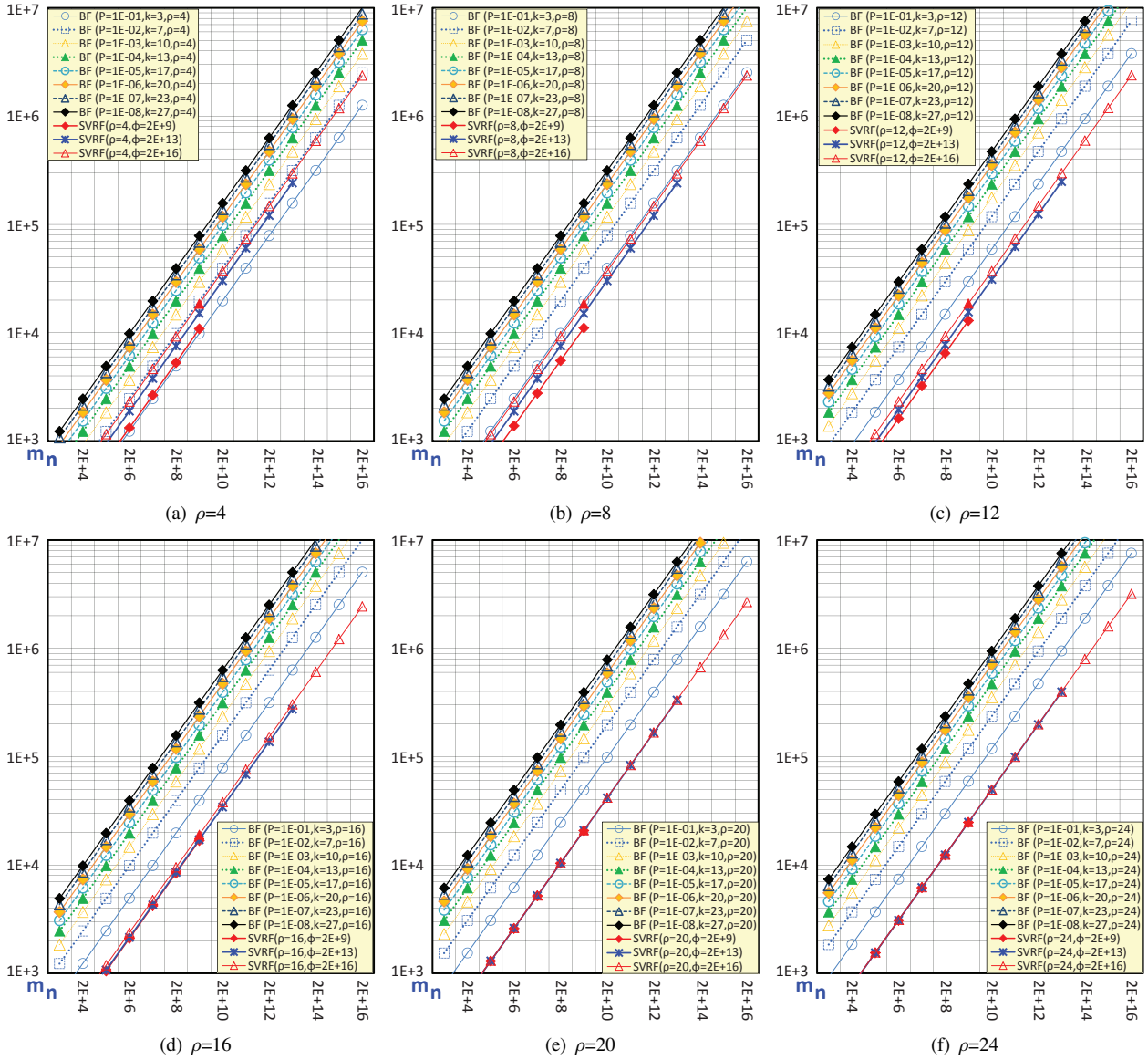
where $\pi(2^\varphi) \leq (n + \pi(2^\rho))$, and $\pi(x) = \frac{x}{\ln x}$ denotes the prime-counting function that gives the number of primes less than or equal to x, for any real number x. Thus the average bit length of $\mathbf{M}_{crt}$ (i.e. space complexity) can be expressed as

$$|\mathbf{M}_{crt}| = \frac{\left(\sum_{i=0}^{\gamma-1}(2^i) + (\mathbf{M}_{cp} - 2^\gamma)\right)}{\mathbf{M'}_{cp}}, \tag{18}$$

where $\gamma = \lfloor \log_2 \mathbf{M}_{cp} \rfloor$, which known as the asymptotic law of distribution of prime numbers.

The first simulation scenario is focused on the *Bits Per Entry (BPE)* $\frac{m}{n}$, and we compare the returns to scale at both port capacities (typically 4∼24 ports) and membership capacities (typically $2^9 \sim 2^{23}$ groups) in order to understand the different growth trend of BPE between the two factors. There are n keys which $> 2^\rho$ are randomly selected from ϕ-elements predefined primes library. Fig.2 presents the required BPE of the SVRF with different parameters being taken into account. The SVRF costs 20.68∼51.47 bits per entry for an 4-port switch, and 48.19∼52.76 bits per entry for a 24-port switch, both of them support up to $2^9 \sim 2^{23}$ total groups, respectively.

The increase of required BPEs $\frac{m}{n}$ was accompanied by an increased membership capacities ϕ, larger increases of BPEs were also associated with larger increases of port capacities ρ. This is because the probability of numbers being prime decreases by a predictable amount on a gradual slope the larger number ranges, In other words, the density of primes continuously decreases as the numbers grow. The results suggest that the low port density switch/router employing proposed scheme is more scalabe and efficient than the Bloom filters.



(a) ρ=4 (b) ρ=8 (c) ρ=12

(d) ρ=16 (e) ρ=20 (f) ρ=24

Fig. 3. Required memory spaces m vs. number of groups n in ρ-port switches.

The second simulation scenario is focused on the memory occupation m versus number of flows n, and we compare proposed scheme to BFs with various parameters taken into account. We instantiate respectively from 2^3 to 2^{16} flows for the reference SDN switches. A ρ-port SDN switch based on BF(P, k, ρ) construction is composed of k hash functions with false positive ratio P for each, and ρ-ports implies that BF construct will be duplicated ρ copies themself. We assume the false positive rate of BF is between $1E-1$ and $1E-8$, and the optimal number of the hash functions k is always set. There are n flows from $2^3 \sim 2^{16}$ to be inserted into each BF construction for each port, where m is the total memory occupations of each switch. In contrast, a ρ-port SDN switch based on SVRF(ρ, ϕ) supports a maximum of ρ ports and ϕ flows. Similar to previous results, many flows (MFEs) can lead to larger scalar-pair in proposed scheme. A longer scalar-pair causes a larger memory usage, so reduced time effectiveness of proposed scheme is expected. The simulation results are depicted in Fig.3(a)$\sim$(f) which display results for different port capacities from 4 to 24 respectively. We can observe that the memory occupation of BF(1E-01,3,4) is the only item lower than the SVRF (in Fig.3(a)), but it seems impossible for any application to accept such a high false positive probability. Besides, the memory occupation of the SVRFs are much lower than BFs in all parameters, by more than several orders of magnitude. Therefore the MFT of switch is compressed into much smaller size without any false positive by proposed scheme.

Although the maximum multicast membership (flow) capacity of practical switches currently rarely exceeds 1,500 groups (flows) [51], we cannot rule out this possibility that to face higher membership capacity requirements in the near future. Compared to Bloom filters, the simulation results indicate that SVRF offers remarkable performance in scalability of number of flows in large-scale software-defined networks.

B. Time Efficiency

It's well-known that the average time complexity of inserting a new element to a standard Bloom filter is the same $\mathcal{O}(k)$, the average time complexity of membership queries

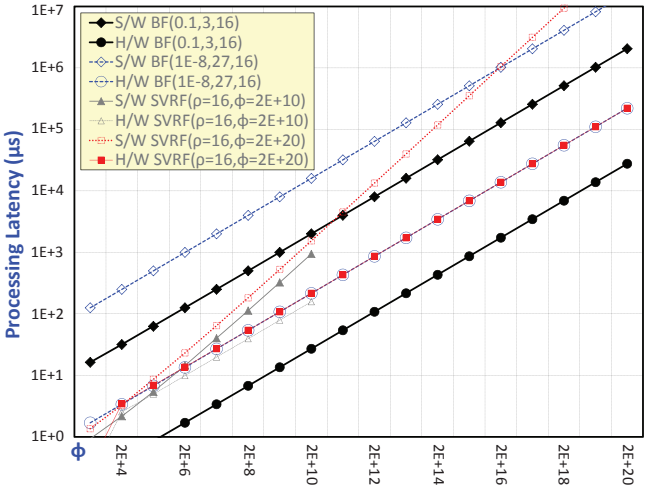


Fig. 4. Packet processing latency vs. membership capacity ϕ.

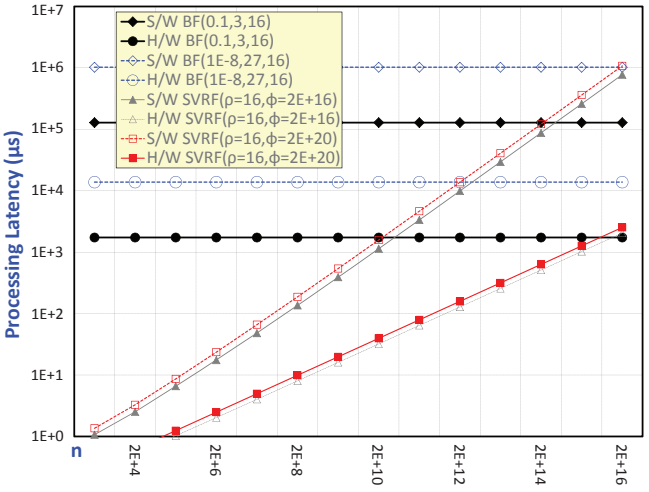


Fig. 5. Packet processing latency vs. number of flows n.

for standard Bloom filters is also $\mathcal{O}(k)$, where k denotes the number of hash functions used by it. In hardware based Bloom filters, The search and insertion run in $\mathcal{O}(1)$ time as with the parallelize hash funcations.

For the proposed scheme, let n be number of MFEs (unicast and multicast flows) which belonging to set A (MFT). The time complexity to construct the scalar-pair of SVRF would be

$$\mathcal{O}\left(n \times \log \mathbf{M}_{cp}\right), \tag{19}$$

note that the $\mathbf{M}_{cp}$ is an intermediate product in constructing the $\mathbf{M}_{crt}$, thus reducing construction time drastically. In this respect, we still not find a way to go beyond the hardware based algorithms such as Bloom filters indeed.

Forwarding latency is the most important metrics for evaluate forwarding performance in a SDN switch, which depends to the time complexity of membership queries of proposed scheme. As mentioned above, the SVRF is an arithmetic approach for multicast forwarding algorithm, so a massive amount of process time will be saved in the SDN switches. Without loss of generality, the proposed scheme can accomplish packet forwarding with very low latency for reasonable size of n (represented by $|\mathbf{M}_{cp}|$+$|\mathbf{M}_{crt}|$). We would expect an even lower packet processing latency for proposed scheme compared to the software based algorithms such as table lookup. Superficially, we can measure the time complexity of proposed algorithm is $\mathcal{O}(1)$ by the number of operations that the algorithm executes. However, when the operands are huge (e.g., 10,000-bit binary string). In such cases, we have to take into account the size of the operands, and to be aware that arithmetic operation is not simple. Proposed algorithm may be inefficient, because the sizes of the operands are ignored.

In the proposed scheme, there are two approaches to accomplish this modulo operation: Software (S/W) and Hardware (H/W) implementations. In software approach, a general-purpose processor is used in the SDN switch. Normally, such platforms have only 32-bit or 64-bit Division (DIV) instruction, thus a modulo operation with long operands is calculated based on multiple divisions, multiplications, and proper truncations using a shorter-length instruction set [52]. The membership query complexity of SVRF is bounded by

$$\begin{aligned}&\mathcal{O}\left(\left(|\mathbf{M}_{cp}| - |k_i| + 1\right) + \left(|\mathbf{M}_{crt}| - |k_i| + 1\right)\right)\\ &\approx \mathcal{O}(2 \times (|\mathbf{M}_{cp}| - |k_i|)),\end{aligned} \tag{20}$$

where $|\mathbf{M}_{crt}|$ denotes the bit-length of $\mathbf{M}_{cp}$ at the current switch, and $|k_i|$ denotes the bit size of key i. We have a improved algorithm grows like $O(n^{\log_2 3}) \approx O(n^{1.58})$ with the digit n, by using $n/2$-digit instruction [53]. Assuming we have a 1024-bit $\mathbf{M}_{cp}$ and $\mathbf{M}_{crt}$, it need 485 multiplications ($\approx$7.275μs) to compute the remainder recursively, by using a 32-bit processer with 15-cycle Multiply (MUL) instruction (ignoring memory access time).

Regarding the bit length of keys $|k_i|$, each multicast group (flow) must be assigned a unique key using different approach in a SDN switch. Since there are one million prime numbers with 24 bits, which thus can represent the keys for all groups in ϕ, and no more than 32 bits for future extension.

In H/W approach, A hardware accelerator for solving very long integer division or modulo operations, based on either FPGA or ASIC [54], is used by SVRF-enabled SDN switch. We have the time complexity for hardware-assisted processing of proposed scheme is bounded by

$$\mathcal{O}\left(\left|\frac{\mathbf{M}_{cp}}{\theta}\right| \times (\delta + o)\right), \tag{21}$$

where θ denotes the θ-bit integer dividers we used, note that $\theta \geq |k_i|$, and δ denotes the number of clock cycles by these dividers, and o denotes the overhead per division such as arithmetic shift times. Most SDN switches use store-and-forward [55] transmission at the input and output of the link, which introduce a transmission latency between each switches along the packet's forward. The packet processing latency of SDN switch should be shorter than average transmission and input/output queuing latencies of the packet. Hence, the processing latency of proposed algorithm would be negligible ($\leq$ tens of cycles).

In this subsection, we compare packet processing performance of the SVRF with BFs. The simulation scenario is as follows: We instantiate increasingly from 2^3 to 2^{20} membership capacity ϕ for a 16-port SDN switch. The switch

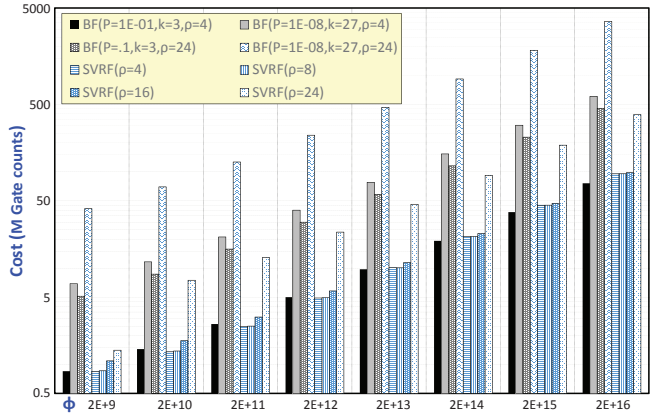


Fig. 6. Hardware costs comparison

have ϕ flows simultaneously active, that's mean the switch is in the n=ϕ condition. Each group randomly select average 8 outgoing ports from the combination of total 16 ports. In S/W-implementation, we assume that each random memory access in each switch requires 50ns by DRAM, and the computing power of each switch to be fixed at a 1GHz 32-bit microprocessor, which supports 15-cycle integer modulo instruction. Each link has sufficient bandwidth and non-blocking switch capacity to accommodate the traffic need in the reference network. In H/W-implementation, we consider that both the hardware accelerator of BF and SVRF are built using the commercial 1GHz 32-bit SRAM-based FPGA, which the datapath between memory and processing unit has a width of 32 bits. We assume that each built-in SRAM and DRAM access in the FPGA requires 10ns and 50ns respectively, and each comparison operation requires 1 cycle.

Fig.4 presents the average packet processing latency experienced by the reference switch versus the maximum membership capacity ϕ at steady state in the reference network. We compare both H/W and S/W-implemented SVRF(16,2E-10) and SVRF(16,2E-20) to BF(0.1,3,16) and BF(1E-08,27,16) as described above, which depicts the low-bound and up-bound of compared schemes. Although the H/W BF(0.1,3,16) has an advantage over any existing schemes, note that it is almost useless to the most applications. The curves show that H/W SVRF(16,2E-10) can offer better performance than H/W BF(1E-08,27,16) schemes, and the H/W SVRF(16,2E-16) also offer similar (indeed better) performance than H/W BF(1E-08,27,16) schemes in the packet processing latency. The curves also show that S/W SVRF is much efficient than all S/W BF schemes in the packet processing latency with no more than 2^9 groups.

Fig.5 presents the average packet processing latency experienced by the reference switch versus the number of flows n, where the maximum membership capacity is fixed at 2^{16}. In order to execute a membership query, BF must scan the whole predefined memory space to compare each bit with the hash. Hence it costs a constant time to determine the final results. Due to the memory usage of SVRF is progressively increases, thus it is not necessary to access full-scale ranges of memory blocks when only part of the memory is in need of calculating the scalar-pair. The results fully demonstrate

TABLE I
THE MAJOR COMPONENTS USED FOR COMPARED SCHEMES

Components	SVRF	BF
DRAM($\mathbf{C_{dram}}$)	4	N/A
SRAM($\mathbf{C_{sram}}$)	6	6
Integer Divider($\mathbf{C_{div}}$)	10–20K	N/A
Hash($\mathbf{C_{hash1}}$)	N/A	1–2K
Prime Hash($\mathbf{C_{hash2}}$)	5–10K	N/A
Multiplex($\mathbf{C_{mux}}$)	1K	0.5K

that our scheme is effectively reduces latency in multicast routing and forwarding of the SDN switches. Indeed, proposed algorithm have become less efficient of time in ultra-scalable routing/forwarding engines (i.e., $\phi(n) > 2^{16}$). because the characteristic of prime number. We still not find a way to go beyond this hurdle, neither Bloom filter and its variants.

The only drawback of the proposed scheme is about its time efficiency. We expect that a lot of questions will be raised on the time complexity of our proposed scheme, this task is also a typical topic for calculating very large numbers in computational number theory [56]. Nevertheless, based on Moore's law, we believe that the performance of 'very long integer divider' such as 8192b/4096b hardware divider would become feasible eventually, if network industries in need of it. And it could be deployed to achieve a routing and forwarding decision in a few clock cycles.

C. Hardware Costs

In this subsection, we estimate the efficiency of proposed scheme by counting and comparing the hardware costs with Bloom filter. Table.I summarizes the major components and their costs (unit: gate counts) required by the compared schemes. We then describe the cost models that are utilized to estimate the hardware costs. By Table.I, the total cost function of Bloom filter can be formulated as

$$\mathbf{C}_{bf} = \sum_{i=1}^{\rho}\left(\sum_{j=1}^{k}\left(\mathrm{C_{hash1}}\right) + \sum_{n=1}^{m(bf)}\left(\mathrm{C_{sram}}\right) + \mathrm{C_{mux}}\right). \tag{22}$$

The total cost function of the proposed scheme is defined as

$$\mathbf{C}_{svrf} = \left[\begin{array}{c}\sum_{i=1}^{m(svrf)}\mathrm{C_{dram}} + \sum_{i=1}^{\nu}\mathrm{C_{sram}} \\ +\mathrm{C_{hash2}} + 2\mathrm{C_{id}} + \mathrm{C_{mux}}\end{array}\right], \tag{23}$$

where ν denotes the minimum cache memory requirements depending on the speed of integer dividers and access time of DRAM. $m(bf)$ and $m(svrf)$ denote the memory usages in BF and SVRF respectively, which reported in the previous subsection. By using the cost models, Fig.6 shows the comparison results of compared schemes. The costs of compared schemes are in direct proportion to their maximum membership capacities ϕ and maximum port capacities ρ, regardless of their manufacturing costs. Observably, the total hardware costs of BF are much greater than proposed scheme's in any conditions, especially for achieving low false positive probabilities.

V. Conclusion

Multicast routing algorithm and its theoretical results have continued to be a very important research topic in the areas of networking. In this research area, Bloom Filter is an efficient algorithm to compress the multicast forwarding states, but significant traffic leakage may occur when group membership querying is false positive. In this paper, we present a novel unified unicast and multicast forwarding algorithm based on the prime number and Chinese remainder theorem in SDN-based datacenter networks, to overcome the scalability problems in terms of multicast forwarding states. We prove that this traditional method can efficiently resolve the exact multiple membership problem. The simulation results indicate that our proposed scheme offers remarkable performance in reducing the memory space, processing time and hardware cost compared with the general Bloom filter. It is also worth noting that scalability and 100% accuracy of proposed scheme are beyond comparison to standard Bloom filter.

The SDN switches used in today's datacenter networks do not scale well with increasing forwarding states and speed. Rather than continuing to scale the Bloom Filters to theirs limit, or build SDN switches with ever larger and faster memory such as CAM in the control plane. Our proposed scheme provides an alternative solution to effectively support the growing demands of multicast applications in managing the software-defined datacenter networks. Network customers, vendors and administrators will thus be benefited by the proposed scheme. We believe that future SDN switches should leverage the proposed scheme as a more scalable and cost-effective solution. In addition, the work in this paper can be extended for different applications. Apart from improving the performance of network routing, it can be extensively applied in many functions of computer science which are dominated by the Bloom filter previously.

References

[1] A. Benslimane, *Multimedia Multicast on the Internet*. London, UK: ISTE, Wiley, 2010.

[2] A. Bianco, P. Giaccone, E. M. Giraudo, F. Neri, and E. Schiattarella, "Multicast support for a storage area network switch," in *GLOBECOM*, Nov. 2006, pp. 1–6.

[3] Y. Vigfusson, H. Abu-Libdeh, M. Balakrishnan, K. Birman, R. Burgess, G. Chockler, H. Li, and Y. Tock, "Dr. multicast: Rx for data center communication scalability," in *Proc. 5th European conference on Computer systems*, ser. EuroSys '10. New York, NY, USA: ACM, 2010, pp. 349–362.

[4] S. Keshav and R. Sharma, "Issues and trends in router design," *Comm. Mag.*, vol. 36, no. 5, pp. 144–151, May 1998.

[5] H. Holbrook, B. Cain, and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast," RFC 4604, Internet Engineering Task Force, Aug. 2006.

[6] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "Esm: efficient and scalable data center multicast routing," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 944–955, Jun. 2012.

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

[8] C. Ding, D. Pei, and A. Salomaa, *Chinese remainder theorem: applications in computing, coding, cryptography*. River Edge, NJ, USA: World Scientific Publishing, 1996.

[9] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.

[10] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 131–145, Oct. 2001.

[11] R. Callon and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)," RFC 4110, Internet Engineering Task Force, Jul. 2005.

[12] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Comm. Mag.*, vol. 47, no. 7, pp. 20–26, Jul. 2009.

[13] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 75–86, Aug. 2008.

[14] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," *Commun. ACM*, vol. 54, no. 3, pp. 95–104, Mar. 2011.

[15] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, Aug. 2009.

[16] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, Aug. 2009.

[17] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," in *Proc. 8th ACM SIGCOMM Workshop on Hot Topics in Networking (HotNets)*, Oct. 2009.

[18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[19] J. Moy, "Multicast Extensions to OSPF," RFC 1584, Internet Engineering Task Force, Mar. 1994.

[20] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An architecture for wide-area multicast routing," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 126–135, Oct. 1994.

[21] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," RFC 1075, Internet Engineering Task Force, Nov. 1988.

[22] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (cbt)," *SIGCOMM Comput. Commun. Rev.*, vol. 23, no. 4, pp. 85–95, Oct. 1993.

[23] 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.

[24] D. Thaler and M. Handley, "On the aggregatability of multicast forwarding state," in *Proc. 21th IEEE INFOCOM*, Apr. 2000, pp. 1654–1663.

[25] F. Németh, A. Stipkovits, B. Sonkoly, and A. Gulyás, "Towards smartflow: case studies on enhanced programmable forwarding in openflow switches," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 85–86, Aug. 2012.

[26] W. chang Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "Stochastic fair blue: A queue management algorithm for enforcing fairness," in *Proc. 21th IEEE INFOCOM*, Apr. 2001, pp. 1520–1529.

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

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

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

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

[31] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal, "The bloomier filter: an efficient data structure for static support lookup tables," in *Proc. 15th annual ACM-SIAM symposium on Discrete algorithms (SODA '04)*, Philadelphia, PA, USA, Jan. 2004, pp. 30–39.

[32] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, Jun. 2000.

[33] M. Mitzenmacher, "Compressed bloom filters," *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 604–612, Oct. 2002.

[34] R. P. Laufer, P. B. Velloso, and O. C. M. B. Duarte, "A generalized bloom filter to secure distributed network applications," *Comput. Netw.*, vol. 55, no. 8, pp. 1804–1819, Jun. 2011.

[35] Y. Hua, Y. Zhu, H. Jiang, D. Feng, and L. Tian, "Supporting scalable and adaptive metadata management in ultralarge-scale file systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 4, pp. 580–593, Apr. 2011.

[36] A. Kumar, J. Xu, and J. Wang, "Space-code bloom filter for efficient per-flow traffic measurement," *IEEE J. Sel. A. Commun.*, vol. 24, no. 12, pp. 2327–2339, Dec. 2006.
[37] S. Cohen and Y. Matias, "Spectral bloom filters," in *Proc. 2003 ACM SIGMOD international conference on Management of data (SIGMOD '03)*, Jun. 2003, pp. 241–252.
[38] D. Guo, J. Wu, H. Chen, Y. Yuan, and X. Luo, "The dynamic bloom filters," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 1, pp. 120–133, Jan. 2010.
[39] D. Guo, H. Chen, and X. Luo, "Theory and network applications of dynamic bloom filters," in *Proc. 25th IEEE INFOCOM*, Apr. 2006, pp. 1–12.
[40] A. Kirsch and M. Mitzenmacher, "Distance-sensitive bloom filters," in *Proc. 8th Workshop Algorithm Eng. and Experiments (ALENEX '06)*. SIAM, Jan. 2006, 0898716101.
[41] B. Xiao and Y. Hua, "Using parallel bloom filters for multiattribute representation on network services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 1, pp. 20–32, Jan. 2010.
[42] H. Song, F. Hao, M. S. Kodialam, and T. V. Lakshman, "Ipv6 lookups using distributed and load balanced bloom filters for 100gbps core router line cards," in *Proc. 28th IEEE INFOCOM*, Apr. 2009, pp. 2518–2526.
[43] F. Hao, M. S. Kodialam, T. V. Lakshman, and H. Song, "Fast multiset membership testing using combinatorial bloom filters," in *Proc. 28th IEEE INFOCOM*, Apr. 2009, pp. 513–521.
[44] F. Hao, M. S. Kodialam, and T. V. Lakshman, "Incremental bloom filters," in *Proc. 27th IEEE INFOCOM*, Apr. 2008, pp. 1067–1075.
[45] O. Rottenstreich, Y. Kanizo, and I. Keslassy, "The variable-increment counting bloom filter," in *Proc. 31th IEEE INFOCOM*, Apr. 2012, pp. 1880–1888.
[46] *IEEE 802.1AB Station and Media Access Control Connectivity Discovery*, IEEE Std., May 2005. [Online]. Available: http://standards.ieee.org/getieee802/download/802.1AB-2005.pdf
[47] S. Bhattacharyya, "An Overview of Source-Specific Multicast (SSM)," RFC 3569, Internet Engineering Task Force, Jul. 2003.
[48] H. Miyatake, M. Tanaka, and Y. Mori, "A design for high-speed low-power cmos fully parallel content-addressable memory macros," *IEEE J. Solid-State Circuits*, vol. 36, no. 6, pp. 958–968, Jun. 2001.
[49] A. J. McAuley and P. Francis, "Fast routing table lookup using cams," in *Proc. 12th IEEE INFOCOM*, Apr. 1993, pp. 1382–1391.
[50] J. Wade and C. Sodini, "A ternary content addressable search engine," *IEEE J. Solid-State Circuits*, vol. 24, no. 4, pp. 1003–1013, Aug. 1989.
[51] D. Newman, "10 gig access switches: Not just packet-pushers anymore," *Netw. World*, vol. 25, no. 12, pp. 34–39, Mar. 2008.
[52] D. E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing, 1997.
[53] A. A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Soviet Physics Doklady*, vol. 7, pp. 595–596, 1963.
[54] M. Lu, *Arithmetic and Logic in Computer Systems*. Hoboken, New Jersey: Wiley-Interscience, 2004.
[55] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks*, vol. 3, no. 2, pp. 97–133, Jun. 1973.
[56] V. Shoup, *A computational introduction to number theory and algebra*. New York, NY, USA: Cambridge University Press, 2005.

Wen-Kang Jia (S'09–M'11) received his 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. His recent research interests are the OSI Layer2∼5 such as TCP/IP protocols design, mobile management, error resilience coding, multimedia communications, NAT traversal, routing and switching, multicasting and broadcasting, teletraffic engineering, IP-optical convergence networks, P2P overlay networks, and wireless networks. He has published over 15 journal articles, over 30 international conference papers, 3 book chapters, and 3 patents. His work has been cited more than 300 times.

Li-Chun Wang (M'96–SM'06–F'11) received the B.S. degree from National Chiao Tung University, Taiwan, R. O. C., in 1986, the M.S. degree from National Taiwan University in 1988, and the Ms.Sci. and Ph.D. degrees from the Georgia Institute of Technology, Atlanta, in 1995, and 1996, respectively, all in electrical engineering. From 1990 to 1992, he was with the Telecommunications Laboratories of the Ministry of Transportation and Communications in Taiwan (currently the Telecom Labs of Chunghwa Telecom Co.). In 1995, he was affiliated with Bell Northern Research of Northern Telecom, Inc., Richardson, TX. From 1996 to 2000, he was with AT&T Laboratories, where he was a Senior Technical Staff Member in the Wireless Communications Research Department. Since August 2000, he has joined the Department of Electrical and Computer Engineering of National Chiao Tung University in Taiwan and is the current Chairman of the same department. His current research interests are in the areas of radio resource management, crosslayer optimized techniques for heterogeneous wireless networks, and cloud computing for mobile applications.

Dr. Wang was elected to the IEEE Fellow grade in 2011 for his contributions in cellular architecture and radio resource management in wireless networks. He won the Distinguished Research Award of the National Science Council, Taiwan, in 2012, and was a co-recipient (with Gordon L. Stüber and Chin-Tau Lea) of the 1997 IEEE Jack Neubauer Best Paper Award for his paper "Architecture Design, Frequency Planning, and Performance Analysis for a Microcell/Macrocell Overlaying System," IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, vol. 46, no. 4, pp. 836848, 1997. He has published over 150 journal and international conference papers. He served as an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2001 to 2005, the Guest Editor of the Special Issue on "Mobile Computing and Networking" for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS in 2005 and on "Radio Resource Management and Protocol Engineering in Future IEEE Broadband Networks" for IEEE Wireless Communications Magazine in 2006. He holds 10 US patents.