Efficient and Adaptive P^{3}FA Forwarding Using Popularity-Based Egress Clustering

Yahui Ding[1], Wen-Kang Jia[1(✉)], and Zhu Jin[2]

[1] College of Photonic and Electronic Engineering, Fujian Normal University, Fuzhou, China
wkjia@fjnu.edu.cn
[2] School of Information Science and Engineering, Southeast University, Nanjing, China
qsx20210831@studuent.fjnu.edu.cn

Abstract. Addressing the issue of conventional forwarding techniques experiencing slow lookup speeds when managing substantial and intricate forwarding tables, which consequently results in increased latency, inefficiency, and suboptimal network resource utilization, this paper proposes a novel forwarding scheme called Egress Clustered P^{3}FA, employing a graph-theoretic clustering algorithm for egress nodes organization based on their popularity. By partitioning the forwarding table into distinct groups, the graph-theoretic clustering algorithm effectively reduces the copious forwarding entries to a smaller and more manageable table. We exploit the correlations among ports to identify output port clusters exhibiting high and low interrelatedness. Simulation results demonstrate that our proposed method significantly enhances the forwarding lookup speeds and optimizes network resource utilization. With the implementation of Egress Clustered P^{3}FA, a more efficient and stable network transmission is achieved, laying the foundation for future applications and scenarios demanding greater network loads.

Keywords: multicast · graph theory · egress clustering · popularity

1 Introduction

As the rapid advancement of internet and cloud computing technologies continues to propel data center network traffic demands, increasingly stringent requirements are imposed upon the network performance and scalability of data centers [1]. A pursuit of higher efficiency and lower latency levels is necessitated to meet these demands. Conventional packet forwarding techniques may exhibit performance bottlenecks when handling high traffic volumes and dynamic communication across nodes in large-scale networks [2, 3]. Consequently, developing and implementing novel packet forwarding methodologies to enhance data center network performance has emerged as a focal point within the industry.

In multicast communication, the objective is to facilitate the transmission of data packets from the source to the recipients using the proper dissemination based on information available in the multicast routing tables [4]. The maintenance of the multicast

F. Gao et al. (Eds.): Chinacom 2023, LNICST 590, pp. 379–394, 2024.
https://doi.org/10.1007/978-3-031-67162-3_24

routing table, which encompasses route information for multicast data packets, is contingent upon dynamic routing protocols and static route information as configured by network administrators [5]. To enhance the efficiency of multicast forwarding and minimize memory usage, an extensible multicast-enabled algorithm for packet forwarding is considered indispensable for routing tables. This process entails converting information related to the target recipients of each multicast group into forwarding table entries. These entries contain forwarding information such as the next-hop IP address, egress port, and next-hop MAC address, which features in the multicast routing table. In the event of route or topology alterations within the network, IP routing tables are updated, and these changes are subsequently mirrored in the forwarding table [6, 7]. Multicast data packets are dispatched to the recipient(s) through suitable routing and forwarding group deployment. Multicast packets are transmitted to receivers via appropriate routing and forwarding groups, with the Multicast Routing Table (MRT) typically transmuted into the Multicast Forwarding Table (MFT) [8] to aggregate the principal forwarding information created by all routing information protocols. The forwarding engine consults the forwarding table to choose the suitable output interface(s), and the packet data is forwarded to one or multiple interfaces to facilitate multicast operations. Analogous to unicast, the multicast forwarding process necessitates the maintenance of efficacious routing and forwarding tables, which ensures accurate packet delivery to their intended destinations. Nevertheless, excessive routing table sizes might contribute to routing performance decline, ultimately impairing the network's overall performance. To circumvent this issue, it is imperative to optimize routing tables, thus reducing their scale while concurrently guaranteeing the error-free transmission of data packets. The popularity algorithm can be employed to enable concise translation of forwarding tables, further mitigating memory consumption.

Leveraging popularity-based routing constitutes an emerging approach within the domain of network engineering, which optimizes forwarding tables. In this context, the resources and services within a network potentially attain popularity according to the frequency of access or requests. Concurrently, during the forwarding process, network entity ports are clustered based on the interrelations among them, aiming to augment forwarding performance. Identifying clusters with high and low correlations facilitates the simplification of forwarding table structures, thereby reducing forwarding latency and enhancing memory usage efficiency.

In light of this context, we propose the Egress Clustered P^3FA, as an efficient group forwarding technology explicitly tailored for data centers. The Egress Clustered P^3FA enhances network dependability, diminishes latency, and minimizes forwarding table dimensions via the employment of probabilistically optimized path selection. Its underlying design philosophy revolves around optimizing caching system performance, contingent upon the prevalence and access frequency of data. This strategy aims to harness caching resources more effectively, augment hit rates, curtail cache misses, and alleviate request loads [9]. The term "popularity" of data refers to its access frequency, denoting the count of instances accessed or the volume of requests. Data characterized by elevated popularity bears a higher likelihood of being revisited in a brief temporal span, which, in turn, renders it more prone to cache hits. Such an occurrence bolsters both cache hit rates and overall application performance.

We embarked on an exploration of employing various algorithms and data structures, such as the Least Recently Used (LRU) algorithm and the Least Frequently Used (LFU) algorithm [10–12], to estimate and maintain the popularity of data. Concurrently, group forwarding technology, as an essential solution, enables the storage of popular data in expeditiously accessible locations (e.g., memory) for swift retrieval, circumventing incessant acquisition from slower backend storage systems (e.g., databases); hence, we determine cache strategies based on data popularity. Consequently, we adapt caching strategies based on data popularity. Graph-theoretic clustering algorithms facilitate the determination of content caching probabilities according to content popularity and routing information, caching popular data in high-speed storage, thereby circumventing redundant transfer and processing. The performance of this algorithm is evaluated through simulations, which substantiate that caching data in high-popularity high-speed storage can expedite the location and forwarding of requisite data packets, enhancing transmission efficiency and overall performance.

Consequently, this paper proposes an innovative forwarding engine predicated on the concept of high-popularity caching. This engine harnesses the power of graph theory-based clustering algorithms to optimize the routing table. This approach provides a more effective support mechanism for multicast data transmission within large-scale networks.

The subsequent sections of this paper are organized as follows: Section two provides an overview of the current research landscape about high-performance forwarding engines. In Section Three, the Egress Clustered P^{3}FA algorithm is introduced, with a detailed exposition of its operational principles and implementation methodology. Section four entails an analysis and evaluation of experimental results and performance metrics, while Section five concludes the study and offers prospects for future work.

2 Research Problems and Related Works

At present, there is a dearth of systematic research focused on the enhancement of caching strategies. The pivotal concerns are the selection of cache content and the timing of said selection. Traditional routers utilize a Forwarding Information Base (FIB) table to execute routing forwarding functions. However, due to the immense scale of the multicast FIB table, the Packet Forwarding Engine (PFE) is required to perform intricate search operations to locate corresponding multiple egress points, which can precipitate performance bottlenecks.

Jia [13] investigates algorithms pertaining to group members, proposing the deployment of Scalar-pair Vectors Routing and Forwarding (SVRF) as an enhancement for both unicast and multicast packet forwarding within the Packet Forwarding Engine (PFE); the method employs Prime Number Theory for establishing and querying data packet (stream) membership relationships, allocating a unique prime number key to each arriving data stream, with its Residue Number System (RNS) [14] properties encompassing two constituent elements, namely Continued Product (CP) and Chinese Remainder Theorem (CRT) [15]. SVRF can effectively meet the primary requirements of multicast forwarding algorithms, permitting insertion and deletion while maintaining operational simplicity. However, when the port density of the PFE is sufficiently high and a multitude of primes are consecutively multiplied, it results in a considerable scalar pair that

occupies substantial storage space. Consequently, researchers are tasked with proposing a forwarding scheme based on SVRF to address issues of scalability and efficiency enhancement.

Building upon the same concept of SVRF, Zhu Jin [16] introduced the Fractional-N and Scalar-Pair Vectors Routing and Forwarding (Fractional-N SVRF) scheme to reduce storage space and enhance efficiency. The underlying principle of this approach involves randomly dividing all forwarding entries into N groups, generating N relatively smaller scalar pairs as a replacement for the larger scalar pairs produced by SVRF. This allows each partition to operate in parallel, with each partition functioning as an independent SVRF sub-block and executing query operations autonomously. Within each sub-block, the stored scalar pairs {Mcp, Mcrt} are composed of smaller master keys. The scalar pairs of the N sub-blocks perform query tasks concurrently, effectively minimizing memory consumption and computational complexity.

In response to unicast and multicast flows characteristic of sparsely populated ports, Jin [17] introduced the Per-Port Prime Filter Array (PFA) algorithm. Within the P^3FA framework, a substantial membership is partitioned into ρ subdivisions, wherein ρ signifies the port density of the PFE. Consequently, we are left with a comparatively smaller subarray of scalar pairs $\{M_{CP1}, M_{CP2},\ldots, M_{CP\rho}\}$ instead of a single, large scalar pair, M_{CP}, within the memory unit. Each port corresponds to an extensive integer divisor. Leveraging the properties of the Prime Number Theorem, the computation results of M_{CP} divided by various keys yield zero and non-zero outcomes. These results are primarily utilized to ascertain whether a data packet should be forwarded from the respective port. Herein, M_{CP} is the continuous product of k_i.

$$Mcp = \prod_{i=1}^{n} (ki), ki \in K. \tag{1}$$

Contrary to the SVRF and Frac-N SVRF, where each subblock comprises two stored scalar pairs (M_{CP} and M_{CRT}), each subblock in PFA encompasses a solitary stored scalar, M_{CP}. Within SVRF and Frac-N SVRF, each multicast forwarding entries (MFE) employs a relatively lengthier $\rho + 1$ bit key, whereas each unicast forwarding entries (UFE) utilizes a comparatively shorter $[\log_2(\rho + 1)]$ bit key. Upon the arrival of packet x at the input unit of the PFA, the parser initially acquires the identifier of the packet (such as the destination IP address). Subsequently, the prime hash function generates a prime key k_x uniquely corresponding to the packet identifier, ensuring the same packet identifier always maps to the same key. The key k_x is then transmitted to the dividers corresponding to all ports, excluding the input port, for modulus computation (the input port is disallowed from re-forwarding to prevent looping). Ultimately, by amalgamating the computational outcomes of the individual dividers, the packet's OPB is procured and the switch card is notified to complete the forwarding operation. During the query phase of packet x, its OPB is obtained through the following method:

$$\begin{aligned} b_x &= \left[b_{x(1)}, b_{x(2)}, \ldots, b_{x(\rho)}\right] \\ &= \sim \left(\left[M_{cp(1)}, M_{cp(2)}, \ldots, M_{cp(\rho)}\right] mod k_x\right) \end{aligned}. \tag{2}$$

Herein, "~" stands for the inverse operation, which transforms a "non-zero" value into a "0-bit" and a "zero" value into a "1-bit". It should be noted that the OPB is calculated

separately within each subblock of the PFA, and ultimately, the computed values from each subblock are consolidated to derive the final result.

In the context of a single multicast routing forwarding node, the frequency (or probability) with which each port is designated as a forwarding outlet by multicast data packets (streams) can be perceived as its popularity. We take into account that ports with high popularity have a higher probability of being selected by a relatively larger number of data packets (streams), while ports with low popularity are correspondingly less likely to be chosen as forwarding outlets. We consider a similar principle: During the forwarding process, ports with high popularity may have a higher probability of being concurrently selected by a relatively larger number of data packets (streams). In other words, there may exist a specific correlation coefficient among ports with high popularity, such that when port A is selected, ports B or C also have a higher probability of being selected simultaneously. This correlation coefficient could potentially be negative—i.e., a negative correlation: when port A is selected, ports B or C have less chance of being selected simultaneously, also known as a mutually exclusive relationship. To leverage this feature of the correlation coefficient, we extend the single physical port of the original P^{3}FA to a virtual OPB group and introduce an optimization algorithm to cluster (group) the ports with high popularity, thereby reducing memory usage. The design of a high-speed forwarding cache mechanism based on popularity can determine the grouping relationship of high-popularity ports based on the popularity and correlation coefficient, cache the most popular port numbers into the high-speed cache's VOPBG, thus enabling the prime number theory-based P^{3}FA to obtain multiple outlets in a single hit calculation, eliminating the need for repeated individual collection member identification for each outlet, while simultaneously reducing the demand for time (calculation) and space (memory).

To further enhance the efficiency of caching, we employ a graph theory-based clustering (grouping) algorithm to optimize the output port bitmap (OPB) table in this article. In traditional topology models, multicast trees are typically composed of numerous nodes exhibiting low egress-diversities, representing the desired egresses in the packet forwarding engine (PFE) in a sparse manner within the output port bitmap (OPB) [18]. This algorithm analyzes the egress distribution characteristics and egress-diversities concepts of multicast trees, constructing several virtual OPB groups (VOPBG). The P3FA method, by identifying the membership of VOPBG, consequently improves multicast forwarding efficiency and scalability.

To augment the efficiency of our approach, we incorporate the notion of egress-diversities [19]; a concept rooted in graph theory, wherein egress-diversities refer to the out-degree of the Multicast Forwarding Table (MFT) in a specified Packet Forwarding Engine (PFE). More specifically, it denotes the concurrent number of exits for multicast prefix entries at a given node within a particular time interval. This concept provides us with a robust framework to enhance the modeling of Output Port Bitmap (OPB) tables and subsequently optimize them via clustering algorithms. This optimization ensures nodes with high popularity and correlation coefficients are stored more efficiently in the cache. The algorithm, through a detailed analysis of the exit distribution characteristics and the concept of egress-diversities in the MFT, allows for the desired exits to be sparsely represented in the OPB within the PFE, thereby expediting the determination of the output port during the multicast packet forwarding process.

The algorithm in question determines whether individual cache nodes can interconnect to form a cache cluster (complete subgraph) based on their respective weights. To be more precise, this study regards cache nodes as vertices and their interconnections, determined by traffic weight, as edges. Subsequently, we employ a graph connectivity algorithm to ascertain which nodes can be interlinked to form a complete subgraph. This methodology maximizes the utilization of the storage capacity of the cache nodes, enhancing the efficacy of caching. This research paves the way for novel approaches and directions in the evolution of high-speed caching.

3 Egress Clustered P^{3}FA

This section provides a comprehensive discussion on Egress Clustered P^{3}FA. Egress Clustered P^{3}FA employs a clustering algorithm grounded in graph theory to optimize the forwarding table of the Output Port Bitmap (OPB). This methodology enables the representation of large routing tables in an extremely compact form, facilitating swift searches using minimal memory, thereby enhancing the forwarding efficiency and throughput of multicast packets. This study involves modeling the OPB table and translating the data within the OPB into nodes of a graph. Subsequently, by calculating the distance and weight between nodes, we determine the data to be stored in the cache and the data necessitating deletion.

3.1 Construction of Egress Clustered P^{3}FA

The functional diagram of the primary components of Egress Clustered P^{3}FA is depicted in the figure. The main components comprise 1) the input/output units, 2) storage units, and 3) processing units. The Packet Forwarding Engine (PFE) determines the exit direction of data packets by executing forwarding instructions. These instructions are controlled by higher-level components such as the Routing Engine (RE), through dynamic routing protocols, and the Routing Information Base (RIB). The parser of the input unit is employed to capture and parse incoming data streams, extracting stream identifiers and mapping a unique prime key based on prime hashing. Each port corresponds to a lengthy integer division device, used to compute the modulus of the M_{CP} and the key. The M_{CP} is stored in the storage unit as the dividend, while the key, generated from Prime Hash, acts as the divisor. The calculation result, either zero or non-zero, is used to judge whether the packet is forwarded at that port. Based on the computation result, the corresponding output port is first sought in the virtual ports; if not found, it returns to the physical ports for further seeking.

The principal modules of Egress Clustered P^{3}FA predominantly encompass two processing units: 1) the Virtual Port PFA processing unit, and 2) the Physical Port PFA processing unit. Both the virtual and physical output ports are partitioned into ρ sub-blocks, where ρ signifies the port density of the Packet Forwarding Engine (PFE). The focus here is not on the large scalar pair in the memory unit, M_{CP}, but rather a relatively fewer sub-scalar pair $\{M_{CP\,(1)}, M_{CP(2)}, \ldots, M_{CP(\rho)}\}$. Leveraging the properties of the Prime Number Theorem, only the key constituting the M_{CP} can divide the M_{CP}, serving as a criterion to determine whether a given data packet can be forwarded from a particular port.

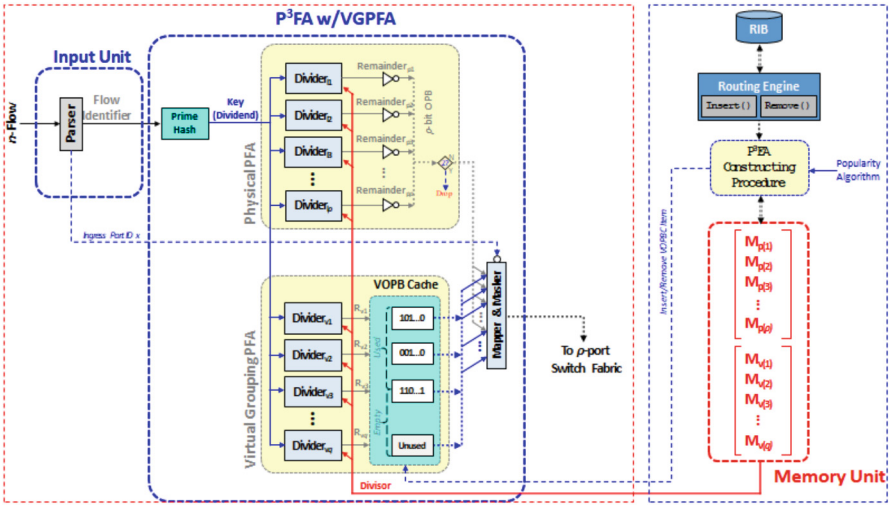


Fig. 1. Egress Clustered P^{3}FA System Block Diagram

3.2 Construction of Forwarding Table

The system functions as a forwarding engine supporting multicast, where the Routing Information Base (RIB) is generated through the learning of dynamic routing protocols [20]. In multicast networks, multicast routing tables document the mapping relationships between multicast sources and groups, as well as the egress interfaces and next-hop information for forwarding multicast data packets. Upon updates to the multicast routing tables, they are transformed into forwarding tables, which guide forwarding decisions when multicast data packets reach the router. The routing engine, in conjunction with the programs constructed in the RIB, converts multicast routing tables into forwarding tables—a process that entails reorganizing routing tables into forwarding tables [21]. Forwarding tables, as an internal data structure within the router, store the forwarding state for each multicast group, indicating whether multicast data packets should be forwarded through each interface, as depicted by the blue dashed line in Fig. 1.

Our research endeavor aims to diminish the space occupied by forwarding tables through the application of a graph theory-based clustering algorithm, where popularity serves as a form of weight. The popularity of a node or edge may be predicated on its frequency of access or use. For instance, a node frequently accessed or utilized will possess a higher degree of popularity. In this manner, we can prioritize nodes and edges with greater popularity, generating the largest complete subgraphs. Our initial step involves the weight calculation for the number of set bits (1s) in each port of the routing table. The recommended principle is to compute the request quantity for each port locally at each node, tallying the weight of the port. We count the frequency of each bit's usage as the node's weight, and when other bits are simultaneously referenced during usage, we consider this frequency as the edge's weight. Based on these weight values, we form an undirected extended weighted graph, denoted as G = (V, E, W1, W2), where V represents the entire port endpoint collection of the Packet Forwarding

Engine (PFE), $|V| = \rho$; E symbolizes the collection of edges between two ports in the Multicast Forwarding Table (MFT) (an edge implies that two ports are simultaneously selected by the MFT as exits); W1 signifies the weight collection of node V (indicating the total number of times a port is selected as an exit by the MFT); and W2 denotes the weight values of collection E (representing the total number of times two ports are simultaneously selected as exits by the MFT). Our algorithm initially generates subsets G1 = {N1}, G2 = {N2}, G3 = {N3},..., Gp = {Nn}, and Gx = {1, 2, 3,..., n} as the basic group POPB, first identifying the node with the maximum weight in Gx and its corresponding edge with maximum weight. Based on the edge with the highest weight, we form a complete subgraph, obtaining an additional group VOPBG, denoted as Gp + 1. We test if Gp + 1 meets the condition of a complete subgraph, specifically, whether all points in Gp + 1 are connected to other points and if the weight is within the range of k. If it does not meet these conditions, we continue our search until we discover a subgraph that does. We then identify the node with the second highest weight in Gx, find the edge with the next highest weight, and sequentially generate Gp + 2, Gp + 3,..., until we find the smallest complete subgraph with two points on a line. In generating each subgraph, we verify whether it meets the conditions of a complete subgraph, continuing our search if it does not until we discover a subgraph that does. Through the aforementioned algorithm, we obtain the additional group VOPBG: Gp + n. According to our proposed clustering algorithm, we can calculate all the output port combinations from the basic group POPB and the additional group VOPBG. Initially, it identifies the popularity of each node and edge in the network by consulting the routing table. Subsequently, based on these popularity measures, it generates the largest complete subgraph and sub-complete subgraph. A complete subgraph is a subgraph in the network where all nodes are connected, whereas a sub-complete subgraph is a complete subgraph minus one or more nodes. In this process, nodes and edges with the highest popularity have the highest priority.

To actualize the above algorithmic of all exit port permutations, we introduce a relaxation parameter k, representing the disparity amidst each port's statistical weights. By harnessing the concept of this "relaxation parameter," said algorithm governs cluster dimensions and adeptly ascertains the optimal subgraph. By fine-tuning this parameter, we establish the generation conditions for the complete subgraphs. As such, the relaxation parameter k functions in this capacity, enabling the identification of all edges with similar weights in the largest complete subgraph based on the weights of each port, sequentially discovering the next largest complete subgraph with similar weights, until no such graph exists.

In summary, the relaxation parameter k serves to control the size of the complete subgraphs. In the generation of complete subgraphs, if the size of the current complete subgraph surpasses the numerical value of the relaxation parameter k set at that time, the search is halted, and the generation of the next complete subgraph commences. This procedure ensures the efficiency and accuracy of the algorithm. The complete subgraphs resemble cache clusters. In this way, we can decompose large OPB tables into numerous smaller cache clusters, thereby reducing computational complexity.

In designing the forwarding table, two optimization objectives were investigated:

1. The difference in port weights - the relaxation coefficient k, which is determined by the weighted statistics of the ports, aims to find the most suitable difference to represent the OPB table with the fewest possible combinations.
2. The maximum allowable number of groups, VOPBG (additional groups) - by comparing the same number of ports, the primary objective is to minimize search time by determining the optimal k value under the same VOPBG.

To evaluate the complete subgraph for the new subnetwork, the algorithmic process is divided into two parts. When two nodes exist in the new subnetwork, the conditions for a complete subgraph are undoubtedly met; when more than two nodes are present, a random node is removed, and the satisfaction of the complete subgraph conditions is assessed. If the conditions are not met, nodes and edges are removed from the subnetwork until additional VOPBG group members satisfying the complete subgraph conditions are found, at which point the search concludes; otherwise, the aforementioned process is repeated. If the node with the highest weight has no neighboring nodes, the subgraph already satisfies the complete subgraph criteria.

3.3 Forwarding Process

In the context of the prime number theorem, the subscale MCP(s) represents the Continued Product (CP) of all keys $k_x \in$ K (integers) and cannot be exactly divisible by any key $k_y \notin$ K, thus resulting in either a hit (zero) or a miss (non-zero) value for each key. Upon the arrival of a packet X at the input unit of the Egress Clustered P^{3}FA, the packet identifier (e.g., destination IP address) is first obtained by the parser. Subsequently, the prime number generator assigns a unique corresponding key k_x based on the packet identifier. Note that the same packet identifier will always map to the same key. The key k_x is then transmitted to all ports (input ports are not allowed to be forwarded again, to prevent loops) of the corresponding division modules 1, 2,…, ρ for modulus operations, as illustrated by the red dashed lines in Fig. 1.

For instance, in the context of an 8-port PFE, when packet x arrives at port 1, the prime hash function allocates a unique key k_x by identifying its packet identifier. Subsequently, k_x is replicated into seven copies (excluding port 1) and dispatched to the respective dividers at ports 2, 3, 4, 5, 6, 7, and 8. The corresponding dividers perform modulus operations, and the packet initially reaches the associated virtual ports, where it undergoes modulus operations with the scalars $\{M_{CP(2)}, M_{CP(3)}, M_{CP(4)}, M_{CP(5)}, M_{CP(6)}, M_{CP(7)}, M_{CP(8)}\}$. The results yield $(M_{CP(2)} \bmod k_x) = 0$, $(M_{CP(4)} \bmod k_x) = 0$, and $(M_{CP(5)} \bmod k_x) = 0$. This implies that packet x is concurrently forwarded to ports 2, 4, and 5, denoted as 01011000. At this point, the search is not yet complete; the packet returns to the corresponding physical ports for further examination. The subsequent findings amount to 00000001, and the final merged output is 01011001.

In instances where the intended target header is not found within the routing cache, indicating an absence of a matching entry in the routing table as the data stream arrives, it can be deduced that it is non-existent in both the Routing Information Base (RIB) and the forwarding table. Consequently, the Packet Forwarding Engine (PFE) discards packets with unrecognized addresses and notifies the routing engine for further processing. Multicast packets necessitate specific routing entries, with the routing of the packets contingent upon the source address, the inbound link, and the multicast destination. All these operations are executed within the distinct multicast code residing in the processor. The multicast routes are stored separately in a dedicated multicast forwarding table. The code inspects whether the destination is multicast. If affirmed, it proceeds to search for multicast routes. On the contrary, it retrieves or constructs the route from the forwarding table.

Addressing the issue of popularity in the forwarding table, we will routinely maintain and update this table to ensure that it only encompasses the most recent and frequently used flow entries. Upon arrival of new flow entries, the Routing Engine (RE) initiates a series of operations to process these entries. Specifically, the RE examines the multicast address of the flow entries, the identifiers for the output ports, and the identifiers for the next-hop nodes before incorporating the entry into the forwarding table. In subsequent forwarding, if a packet's destination identifier matches an entry, the cache cluster can directly retrieve this entry from the high-speed cache, obviating the need to access the routing table, thereby enhancing forwarding speed. Unlike traditional Separate Virtual Routing and Forwarding (SVRF), the output OPB serves as an index for the Egress Clustered P^3FA, entering the forwarding table for retrieval. If a corresponding flow entry is matched, the forwarding action is completed. If not, it reverts to the OPB table for re-examination, whilst updating the forwarding table (insertions, deletions, etc.).Through the utilization of Egress Clustered P^3FA, we can effectively optimize the forwarding table, thereby augmenting the performance of the multicast forwarding engine.

The operational mechanism of Egress Clustered P^3FA incorporates four steps: Initially, the weights of each port are quantified, wherein the weight signifies the port's popularity in multicast transmission, i.e., the frequency of port usage or the magnitude of data flow in multicast communication. Subsequently, ports are segregated into several clusters based on their weights. For each cluster, the ports within are treated as nodes, and the edges between nodes denote the frequency of simultaneous usage of two ports. A subgraph is a collection constituted by a group of ports with high correlation. Each subgraph encompasses several ports, with ports of higher weight assigned to a greater number of subgraphs. The objective of this approach is to group ports with minimal weight differences together, facilitating optimal utilization of cache space, and reducing memory consumption. Next, when a packet arrives, it is prioritized to enter this virtual forwarding table composed of several subgraphs for retrieval. The packet can swiftly identify the appropriate egress port, thereby achieving efficient packet forwarding. Finally, if the corresponding flow entry is not located within the forwarding table, Egress Clustered P^3FA reverts to the OPB table for re-examination and updates the forwarding table. Updates may include the addition of new flow entries, to ensure the packet's forwarding path is recorded in the forwarding table, thereby enhancing the lookup speed for similar packets in the future.

In the ensuing sections, we will delve further into the performance evaluation and practical application scenarios of Egress Clustered P^{3}FA. This encompasses experimental methodologies, performance metrics, analysis of experimental results, and optimization strategies tailored for diverse scenarios. Through these elements, we aspire to furnish our readers with a comprehensive implementation scheme of Egress Clustered P^{3}FA, thereby fostering the advancement of multicast technology.

4 Performance Evaluation

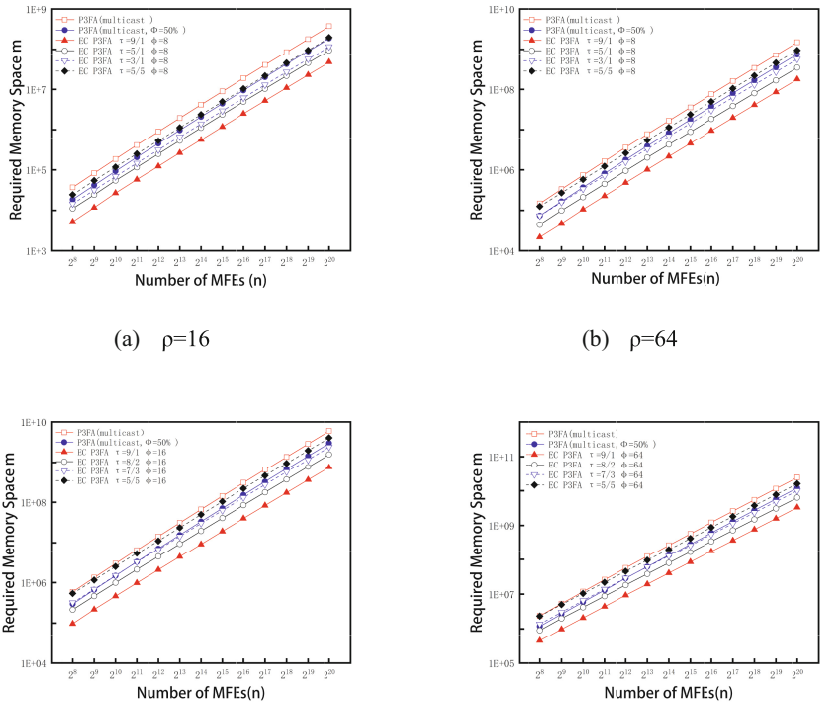
The methodology proposed in this paper involves the construction of a graph-theoretic clustering algorithm in a Python environment and the verification of Egress Clustered P^{3}FA and other competitive schemes in terms of PFE performance under various port densities. During the algorithm simulation, port densities are set from 32 to 1024, with two primary parameters established: the relaxation coefficient 'k' and the additional group VOPBG. This section evaluates the performance of Egress Clustered P^{3}FA by analyzing different values of the relaxation coefficient 'k' and the additional group VOPBG.

Our primary analysis focuses on two metrics: 1) spatial efficiency, and 2) reduction in combination numbers.

4.1 Space Efficiency

Within the processing unit, we have established a scalar array for an ρ-port switch with 'n' entries, represented as $\{M_{CP\,(1)}, M_{CP\,(2)}, \ldots, M_{CP\,(\rho)}\}$. Initially, we generate 'n' groups of ρ-bit binary strings randomly. The data packet of each sub-block will undergo modulus operation with the corresponding sub-scalar M_{CP} stored in the memory unit, obtaining the corresponding OPB according to Eq. (1) to simulate the output port bitmap (OPB) of 'n' multicast streams, i.e., an $n \times \rho$ matrix, where each element is characterized as binary '0' or '1'. Given the existence of numerous repeated output port combinations in the forwarding table, some ports may exhibit correlation, i.e., the probability of multiple ports being selected simultaneously. The higher the correlation, the more frequent the repetition of output port combinations, represented by τ. We observed that a few of the most common combinations can cover all corresponding output ports of the requested data packets. Given this repetitiveness, we generate different OPB tables by setting different egress-diversities (φ values) and correlation τ. For each port density, we employ different φ values to generate multiple mapping rules for output port diversity. The φ value generally ranges from 1 to ρ, and φ follows a normal distribution with a mean of $\rho/2$ and a variance of 1. We simulated the memory usage under different port numbers, specifically 16, 64, 256, and 1024.

The spatial efficiency measures the relationship between the total memory space required, denoted as 'm', and the number of entries in the observed forwarding scheme, represented by 'n'. 'M' represents the overall memory occupancy rate for each PFE. We conducted an in-depth analysis of the memory usage of this approach compared to P^3FA and $P^3FA(\varphi = 50\%)$ under different parameter conditions, such as port density 'ρ', correlation 'τ', and egress-diversities 'φ'. We also performed adaptive adjustments for $\tau = 5/5$, $\tau = 7/3$, $\tau = 8/2$, and $\tau = 9/1$. For each port density, we instantiated varying numbers of forwarding entries, typically ranging from 2^8 to 2^{20}. The simulation results are shown in Fig. 2. The EC P3FA depicted in the figure is an abbreviation for Egress Clustered P3FA.It can be observed that Egress Clustered P^3FA exhibits lower memory consumption compared to the original P^3FA, and as τ increases with fixed φ, the required memory space decreases. This demonstrates that despite introducing a new cache forwarding table, the shorter key length results in lower memory occupancy. Furthermore, even when there is a high correlation among ports, Egress Clustered P^3FA still consumes less memory than the original P^3FA.



(a) ρ=16 (b) ρ=64

(c) ρ=256 (d) ρ=1024

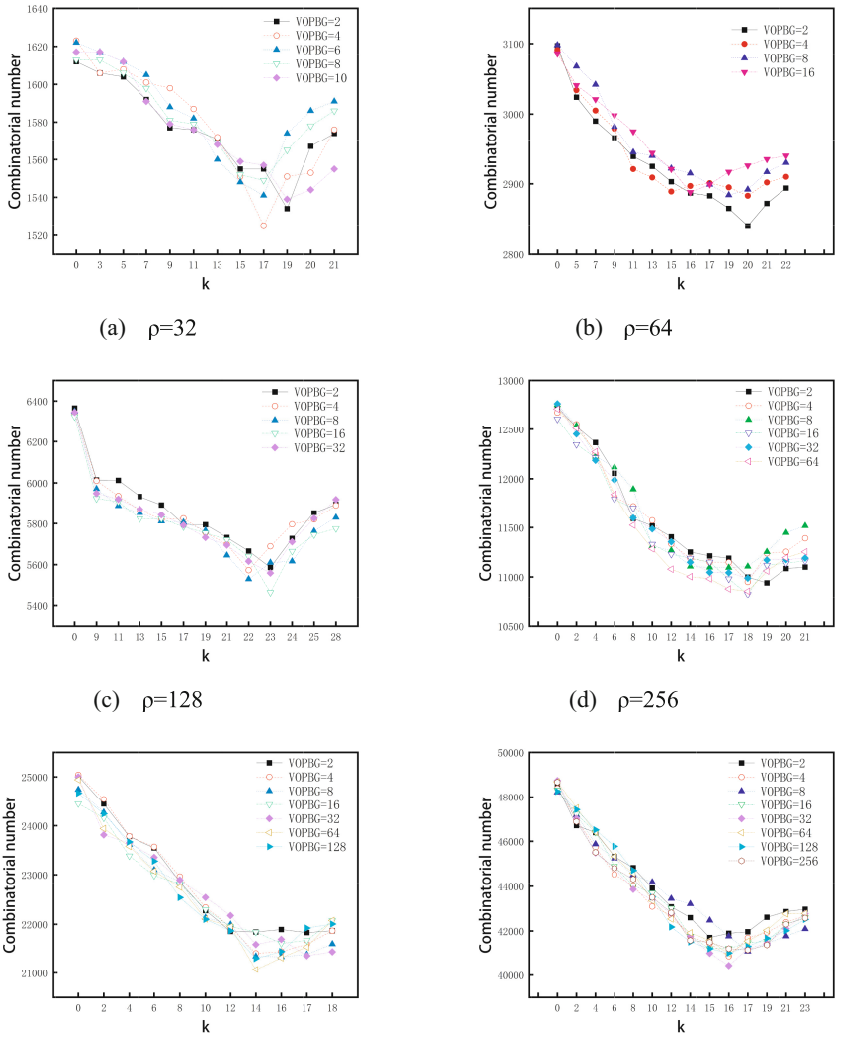
Fig. 2. Forwarding latencies versus number of forwarding entries n in the ρ-port PFEs.

4.2 Comparison of Combination Numbers

The relaxation coefficient 'k' serves as a pivotal factor, directly influencing and controlling the efficiency and precision of the algorithm. Consequently, this experiment examines the performance of the algorithm under different 'k' values, assessing the impact on algorithmic performance by comparing and analyzing the number of combinations at different 'k' values. Under varying port counts, the algorithm's accuracy improves as 'k' increases, but this is accompanied by a corresponding increase in execution time. Furthermore, the optimal selection of 'k' is dependent on the port count, necessitating the choice of the most suitable 'k' value based on specific application circumstances.

The experiment was configured with varying port counts of 32, 64, 128, 256, 512, and 1024. Under these conditions, the number of combinations displays an initial increase followed by a decrease as the 'k' value escalates. When the combinations reach a minimum, it signifies that the prime numbers can be wholly divisible, thereby optimizing the algorithm's execution efficiency. Given the varied port counts, the 'k' value at the optimal number of combinations is not identical; hence, from Fig. 3(a), it can be discerned that when VOPBG = 4, the optimal 'k' value is 17, enhancing the efficiency by 6.03%. Figure 3(b) indicates an optimal 'k' value of 20 when VOPBG = 2, resulting in an efficiency gain of 8.3%. Figure 3(c) reveals that when VOPBG = 16, the optimal 'k' value is 23, leading to an efficiency increase of 12.7%. Figure 3(d) shows that with VOPBG = 16, the optimal 'k' value is 18, improving efficiency by 14.2%. Figure 3(e) demonstrates that when VOPBG = 64, the optimal 'k' value is 14, resulting in an efficiency boost of 15.5%. Finally, Fig. 3(f) shows that when VOPBG = 32, the optimal 'k' value is 16, enhancing efficiency by 16.9%. The figures reveal that for different VOPBG, the number of combinations reaches a minimum with an increasing 'k' value before rising again as 'k' continues to increase. Due to the stochastic generation characteristics of the OPB table, each simulation experiment might yield a different OPB table. This suggests that for different OPB tables, there could be various optimal 'k' values, allowing the number of combinations to reach a minimum. This results in fluctuations in the optimal 'k' value, primarily attributed to the randomness that leads to different combinations of table entries in different simulation experiments. This, in turn, affects the position of the optimal 'k' value. To handle these fluctuations, one might adopt repeated simulations and calculate the average of the results. In this way, across multiple experiments, one can observe the statistical distribution and trends of the optimal 'k' value. This method can, to a certain extent, mitigate the impact of random fluctuations on the simulation results and provide a more stable judgment regarding the optimal 'k' value.

Hence, it can be concluded that as the number of ports increases, the optimal VOPBG value corresponding to the optimal solution also gradually increases. An increase in VOPBG results in an elevated spatial complexity of the algorithm, but it does not impact the algorithm's runtime. Therefore, in situations where sufficient spatial resources are available, it is appropriate to moderately increase VOPBG to enhance the algorithm's accuracy. Simultaneously, under different port count scenarios, the most suitable 'k' values are generally found within the range of 15 to 21. Furthermore, this algorithm can reduce the size of the forwarding table by approximately 6% to 17% while maintaining the network's forwarding performance unaffected. Notably, the performance improvement becomes more pronounced as the number of ports increases.



(a) ρ=32 (b) ρ=64

(c) ρ=128 (d) ρ=256

(e) ρ=512 (f) ρ=1024

Fig. 3. Number of entries with different port combinations

In summary, the method proposed in this study outperforms existing approaches, demonstrating superior performance in obtaining the optimal number of combinations corresponding to the slack coefficient 'k' and the extra group VOPBG under different port count conditions. These two variables can directly influence the performance and efficiency of the algorithm, thereby leading to noticeable performance improvement under optimal conditions. Moreover, when the number of combinations is relatively low, the algorithm exhibits shorter execution times. However, as the number of combinations

increases, the algorithm's execution time escalates correspondingly. This is attributable to the increased number of data packets the algorithm needs to process as the number of combinations rises.

5 Conclusion

Addressing the challenge of optimizing routing table caches, this study proposes a graph-theoretic clustering algorithm-based solution. The Popularity-Based Port Clustering Forwarding (P^{3}FA) represents a breakthrough forwarding mechanism that overcomes the limitations of conventional approaches through real-time analysis of traffic patterns and egress port popularity. By clustering egress ports according to their usage frequency and popularity, forwarding tables are divided into distinct subgraphs. The optimal number of combinations is achieved by setting the slack parameter 'k' and adding extra groups VOPBG, ultimately optimizing routing table caches. In a series of simulation experiments, the proposed algorithm demonstrates superior performance compared to traditional P^{3}FA algorithms, particularly with a significant improvement in the required number of port groups. As the number of ports increases, the performance enhancement becomes more pronounced. Experimental results also reveal the existence of an optimal 'k' value for different port numbers, minimizing the number of combinations and reconstructing the Output Port Bitmap (OPB table). Consequently, this approach meets the requirements for high-performance routing forwarding tables in large-scale packet-switched networks.

In practical applications, the Egress Clustered P3FA algorithm significantly reduces memory consumption of routing tables and exhibits superior performance in combination computations. By setting the 'k' value for weighted subgraph partitioning and introducing additional groups VOPBG, it optimizes large forwarding tables into smaller ones. Experimental results indicate that cache usage performance improves by approximately 15–20%, substantially conserving storage resources. Moreover, as the number of ports increases, the algorithm's performance enhancement becomes more pronounced, adapting to the ever-expanding network scale and data traffic. Consequently, for large-scale packet-switched networks, this graph-theoretic clustering algorithm presents a scientifically reliable and practically feasible solution, offering robust support for network performance improvement and resource optimization.

In summary, our proposed Egress Clustered P^{3}FA strategy enables more judicious utilization of routing table caches, exhibits considerable scalability, and effectively enhances forwarding efficiency. This, in turn, renders the deployment of multicast-supporting routers/switches more efficient and reliable. Future research may delve into applying this method to more complex network topologies and optimizing its efficiency and scalability. Additionally, to further augment router/switch performance, one could consider integrating this algorithm with other optimization techniques, such as optimal path selection, congestion control, and the like.

References

1. Luo, L., Foerster, K.-T., Schmid, S., Yu, H.: Optimizing multicast flows in high-bandwidth reconfigurable datacenter networks. J. Netw. Comput. Appl. **203**, 103399 (2022)
2. Wang, Z., Crowcroft, J.: Shortest path first with emergency exits. In: Proceedings of the ACM Symposium on Communications Architectures & Protocols (1990)
3. Hopps, C.: Analysis of an equal-cost multi-path algorithm. In: Internet Engineering Task Force (2000)
4. Benslimane, A.: Multimedia Multicast on the Internet. John Wiley & Sons, Hoboken (2013)
5. Aweya, J.: Switch/Router Architectures: Shared-Bus and Shared-Memory Based Systems. Wiley, Hoboken (2018)
6. Lin, P., Bi, J., Hu, H.: BTSDN: BGP-based transition for the existing networks to SDN. Wirel. Pers. Commun. **86**, 1829–1843 (2016). https://doi.org/10.1007/s11277-015-3145-0
7. Tsai, P.-H., Tseng, Y.-L., Zhang, J.-B., Tsai, M.-H.: A query-based routing table update mechanism for content-centric network. In: 2020 International Computer Symposium (ICS), pp. 266–271. IEEE, Tainan (2020)
8. Tapolcai, J., Bíró, J., Babarczi, P., Gulyás, A., Heszberger, Z., Trossen, D.: Optimal false-positive-free bloom filter design for scalable multicast forwarding. IEEE/ACM Trans. Netw. **23**, 1832–1845 (2015)
9. Katta, N., Alipourfard, O., Rexford, J., Walker, D.: CacheFlow: dependency-aware rule-caching for software-defined networks. In: Proceedings of the Symposium on SDN Research. pp. 1–12. ACM, Santa Clara (2016)
10. Selvakumar, S., Kumar Sahoo, S., Venkatasubramani, V.: Delay sensitive least frequently used algorithm for replacement in web caches. Comput. Commun. **27**, 322–326 (2004)
11. Gast, N., Van Houdt, B.: TTL approximations of the cache replacement algorithms LRU(m) and h-LRU. Perform. Eval. **117**, 33–57 (2017)
12. Zhang, Z., Wang, B., Lan, J.: Identifying elephant flows in internet backbone traffic with bloom filters and LRU. Comput. Commun. **61**, 70–78 (2015)
13. Jia, W.-K., Wang, L.-C.: A unified unicast and multicast routing and forwarding algorithm for software-defined datacenter networks. IEEE J. Sel. Areas Commun. **31**, 2646–2657 (2013)
14. Molahosseini, A.S., de Sousa, L.S., Chang, C.-H. (eds.): Embedded Systems Design with Special Arithmetic and Number Systems. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-49742-6
15. Pei, D., Salomaa, A., Ding, C.: Chinese remainder theorem: applications in computing, coding, cryptography. World Scientific (1996)
16. Jia, W.-K., Jin, Z.: Fractional-N SVRF forwarding algorithm for low port-density packet forwarding engines. IEEE Netw. Lett. **3**, 42–46 (2021)
17. Jin, Z., Jia, W.-K.: P3FA: unified unicast/multicast forwarding algorithm for high-performance router/switch. IEEE Trans. Consum. Electron. **68**, 327–335 (2022)
18. Deering, S.E., Cheriton, D.R.: Multicast routing in datagram internetworks and extended LANs. ACM Trans. Comput. Syst. **8**, 85–110 (1990)
19. Jin, Z., Jia, W.-K.: DH-SVRF: a reconfigurable unicast/multicast forwarding for high-performance packet forwarding engines. IEEE Trans. Parallel Distrib. Syst. **33**, 1262–1275 (2022)
20. Maltz, D.A., Xie, G., Zhan, J., Zhang, H., Hjálmtýsson, G., Greenberg, A.: Routing design in operational networks: a look from the inside. SIGCOMM Comput. Commun. Rev. **34**, 27–40 (2004)
21. de Morais Cordeiro, C., Gossain, H., Agrawal, D.P.: Multicast over wireless mobile ad hoc networks: present and future directions. IEEE Netw. Netw. **17**, 52–59 (2003)