Fundamentals, Service and Architecture of Programming Protocol-Independent Packet Processors : A Literature Survey

Malldi Saesar Ariffudin [1*], Leanna Vidya Yovita [2], Tody Ariefianto Wibowo [3]
[1,2,3]*Telkom University*
[1,2,3]*Bandung, Indonesia*
Email: mallsa@student.telkomuniversity.ac.id*

ARTICLE INFORMATION

Received 17 January 2025
Revised 17 April 2025
Accepted 21 April 2025

Keywords:
Data Plane
Forwarding
Named Data Networking
Programming Protocol-independent Packet Processors
Software Defined Network

ABSTRACT

The rapid development of network technology necessitates a system capable of adapting to these advancements. Technologies such as 5G, IoT, and Cloud Computing demand flexible and easily configurable network systems. However, the interface differences among providers often hinder this process. Previously, standard connecting protocols were primarily complex and slow to support new protocols. As a solution, P4 emerges as a programming language that allows for the configuration of packet processing and header design according to needs directly in the data plane. This paper discusses the fundamentals of P4 and the services it can run. To clarify its functions, this manuscript explores implementation cases within SDN/IP networks and Named Data Networking (NDN) architecture, including Load Balancing, Caching, Security, Congestion Control, In-band Network Telemetry, Forwarding, and Routing. We also map existing research on each component of SDN and NDN technologies. The manuscript concludes by reviewing challenges and research opportunities for P4 as guidance for further studies.

1. Introduction

The development of information and communication technology (ICT) is rapidly advancing with technologies such as the Internet of Things, 5G, cloud computing, caching, and In-Band Network Telemetry (INT). Referring to an article discussing the hype cycle of networking technologies, advancements such as IoT, 5G, and Cloud Computing are expected to remain trends for development over the next 2 to 10 years (Gartner, 2023). However, configuring these technologies can take time due to the entanglement between the operating system and the data plane and the varying interfaces available for each vendor. To address this issue, a Software-Defined Network (SDN) separates the control plane and the data plane (Kaur et al., 2021). SDN is a centralized network architecture in which one node is a controller that manages forwarding nodes. Therefore, SDN is a flexible, dynamic, and easy-to-manage architecture (Kim & Feamster, 2013). OpenFlow is a standard protocol used for communication between the control plane and the data plane (McKeown et al., 2008).

Initially, OpenFlow interfaces were simple. However, over time, the specifications became increasingly complex, resulting in longer headers. Additionally, OpenFlow has a fixed data plane, which can cause delays when implementing new services or network protocol updates. Programming Protocol-independent Packet Processors (P4) is a high-level programming language that can be used to program the data plane (Bosshart et al., 2014; Budiu et al., 2017).

SDN was developed as a solution to the challenges posed by traditional network architecture in keeping up with rapid and diverse technological advancements. Figure 1 illustrates the evolution from the Down-Top design to the Top-Down design. SDN architecture design is known as Down-Top design, where the data plane remains fixed and cannot be changed. P4 creates a top-down design that allows for the programming of the data

plane and describes how packets are processed, resulting in increased flexibility. P4 capabilities and flexibility enable it to adapt to the development of services in IP or SDN networks, as well as in new networks such as Named-Data Networking. Named-Data Networking is a novel network architecture that prioritizes data or content by assigning unique names to requested data or content, as opposed to IP addressing (Afanasyev et al., 2018; L. Zhang et al., 2014).

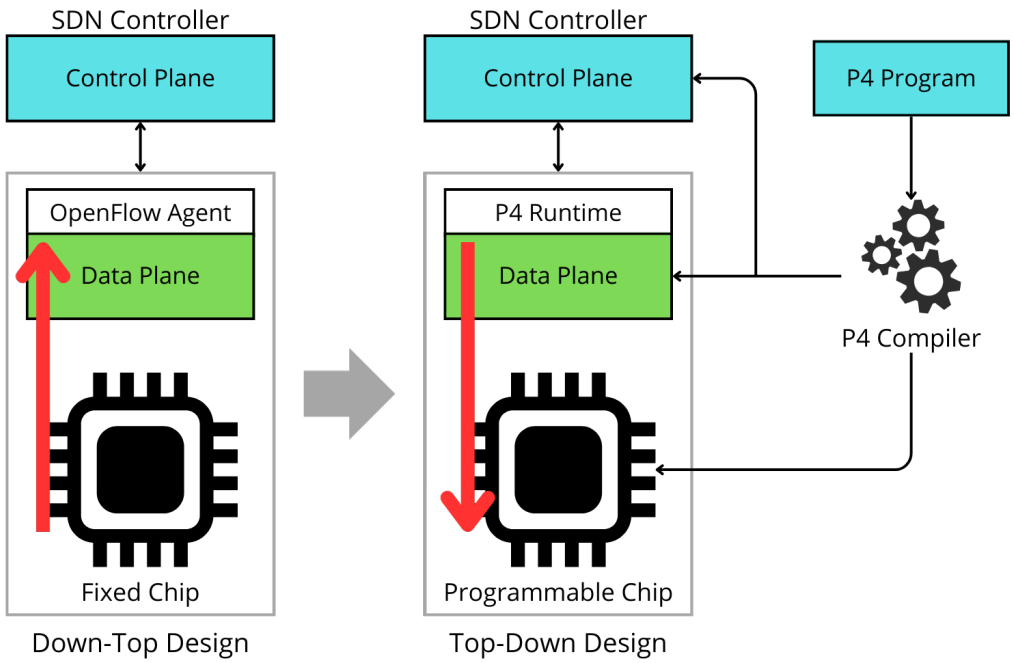

Figure 1. Evolution of Network Architecture

While many excellent survey papers are available, most concentrate solely on the use of P4 in IP/SDN network architecture. However, only a few papers discuss P4 in Named-Data Networking (NDN) (Goswami et al., 2023). This paper maps existing research on foundational protocols (such as OpenFlow, etc.) and P4, focusing on case studies in SDN and NDN. SDN is a technology that separates the control plane from the data plane and is currently widely used to enhance network flexibility and load efficiency. NDN is a content-based technology being developed as a candidate for the future internet, focusing on communication flexibility and efficiency. The paper also discusses P4 in detail, including its working mechanisms and components, along with the advantages it offers. Additionally, the position of P4 is explained within the context of SDN and NDN to provide a clearer picture of its role. Finally, this paper identifies challenges as well as future research opportunities that are important to pursue.

Figure 2 shows that the P4 implementation is divided into two network architectures: Software-Defined Networking and Named Data Networking. Additionally, the compiler is a separate component of P4. SDN is divided into several implementations, including Load Balancing, Caching, Security, Congestion Control, and In-Band Network Telemetry (INT). NDN is divided into three parts: Forwarding, Routing, and Security.

The paper is structured as follows: (ii) This section will describe the P4 program language and architecture, as well as its benefits. (iii) The research conducted on P4, including security, load balancing, caching, and compilers, will be explained. (iv) Open research opportunities and challenges of P4 will be discussed. (v) Finally, conclusions will be drawn. The papers selected for this study were identified using the keywords "Programming Protocol-independent Packet Processors (P4)," "Named-Data Networking," "Software-Defined Network," "P4-NDN," and "Programmable Network".

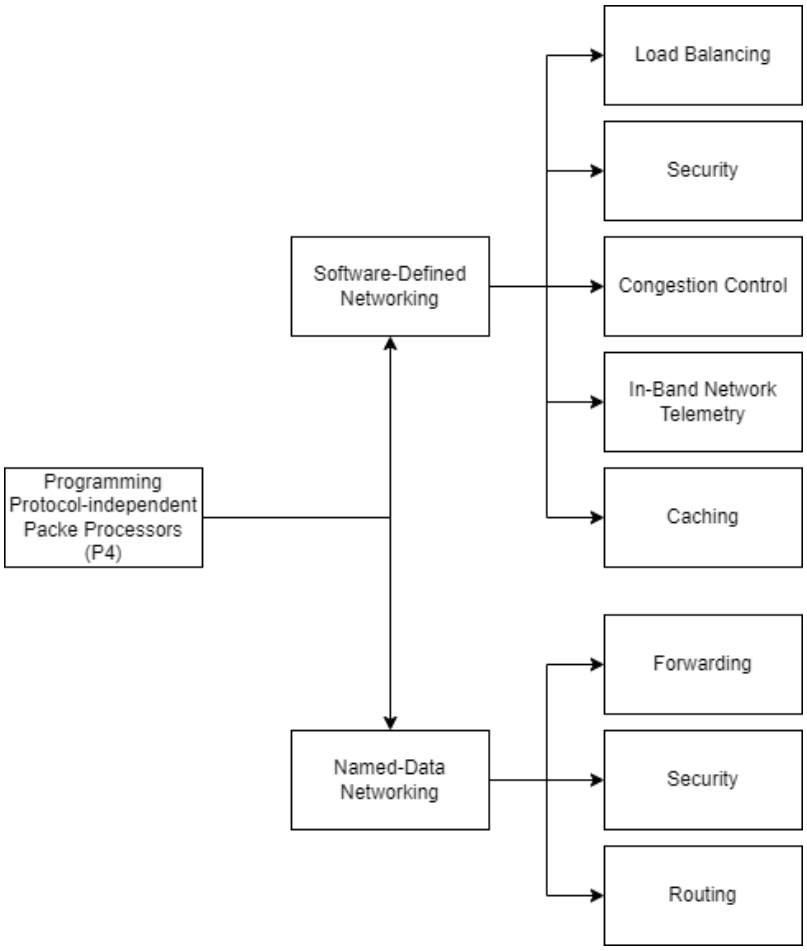

Figure 2. Classification of P4 Implementation

2. Programming Protocol-independent Packet Processors (P4)

P4 is a programming language used to specify how packets are processed by the data plane (forwarding plane). Although it was initially designed for programming switches, it is now used for various purposes. It is important to note that P4 is limited to programming the data plane and cannot be used for the control plane.

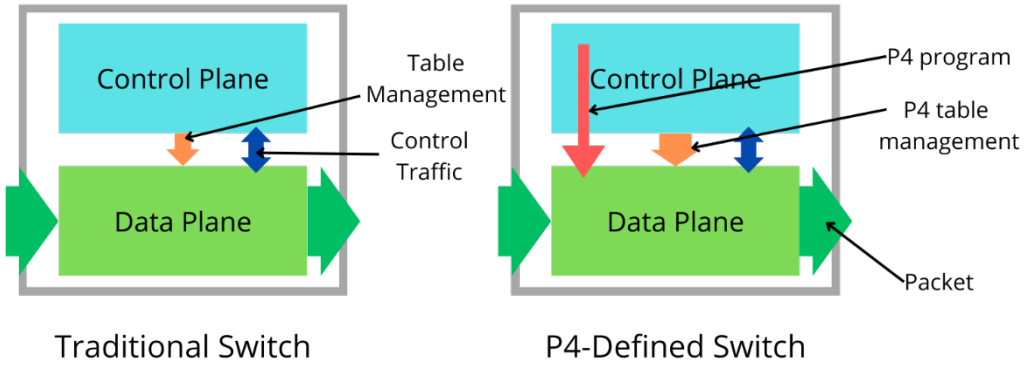

Figure 3. Traditional Switch vs P4-Defined Switch

Figure 3 illustrates the difference between traditional switches and programmable switches (p4-defined switches). In traditional switches, the control plane and data plane are fixed according to the switch chipset and cannot be programmed. However, in programmable switches, the control plane is fixed, and the data plane can be described using the P4 language as needed, even though the communication between the two still uses the same channel.

2.1 *P4 Forwarding Model*

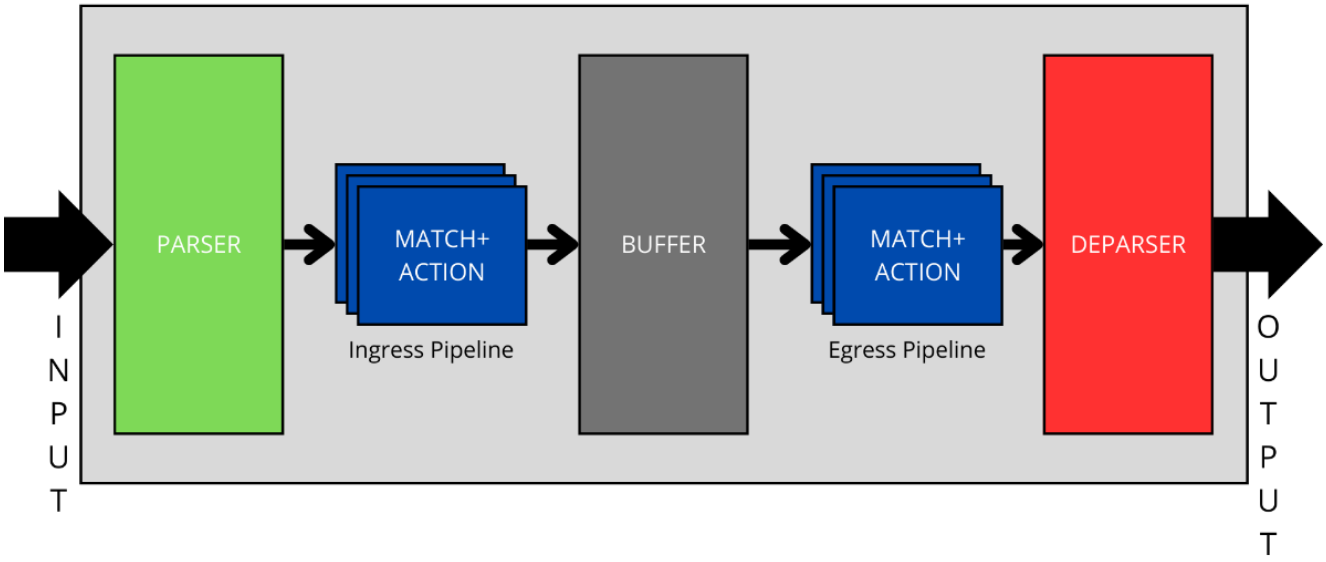

Figure 4. P4 Forwarding Model

The P4 forwarding model comprises multiple function blocks that process incoming packets until they are ready for forwarding, as shown in Figure 4. Each block has a specific packet processing function described in the P4 programming language. Details for the P4 programming documentation can be found in (P4 16 Language Specification Version 1.2.2, 2021)

2.1.1 *Parser*

The parser's task is to extract and recognize each incoming packet based on its header. While this may seem simple, it can be quite complex for certain packets, such as NDN packets, which require a flow of conditions to be added during extraction.

2.1.2 *Match + Action*

After the parser extracts the packet, it is forwarded to match+action. This function determines how the packet will be treated, including whether it will be dropped, forwarded, duplicated, or handled in another way. The match+action function block has two parts: ingress and egress. Although the functions are similar, match+action ingress handles incoming packets while match+action egress handles outgoing packets.

2.1.3 *Deparser*

After the parser extracts and processes the packet using match+action, it is then passed to the deparser function block. The deparser is responsible for reconstructing the packet header that has been processed for forwarding.

2.2 *Defines the Header Format*

The format of the header is a crucial aspect of the P4 program. It defines the structure and width of the packet header, including IPv4, UDP, VLAN, and other protocols. P4's flexibility lies in its capacity to create custom headers and define new packet headers, such as NDN, BPv6, and BPv7.

2.3 *P4 target architecture model*

The P4 architecture is beneficial for describing each function block in the P4 program before implementation on the target. Initially, there was only one P4 architecture, namely the Protocol-Independent Switch Architecture (PISA) in P414. At that time, the available PISA architecture did not include deparsers. However, as P4 developed, P416 now supports the PISA architecture with improvements, as well as other architectures such as the V1Model Switch and Portable Switch Architecture (PSA).

3. Research of P4

Considerable research has been conducted on the P4 programming language, covering topics ranging from its fundamental components, such as compilers and language structures, to its application in emerging services and package types. This section will describe some of the research that has been classified previously, not only within the scope of IP networks but also within the scope of P4 research on NDN networks.

3.1 *P4 Compiler*

The compiler on the P4 is one of the components responsible for translating the program to specific targets and also mapping the targets to particular target switches to generate the appropriate configuration for the device. P4 itself has a default compiler called P4 compiler, but the compiler has limitations for multiple targets. The T4P4S compiler can overcome this limitation, as T4P4S can compile for multiple targets. Furthermore, the T4P4S compiler's results for specific targets demonstrate superior performance compared to OpenvSwitch (OVS) (Vörös et al., 2018). However, the T4P4S compiler has its limitations, as it is still susceptible to numerous bugs and issues in the P4_16 program.

Ibanez et al. developed a compiler designed to overcome the performance limitations of the built-in P4 compiler. By altering the compilation workflow, this compiler aims to produce performance similar to the original devices, but its use will be limited to NetFPGA devices only (Ibanez et al., 2019). On the other hand, Zhu et al. introduced a compiler that supports virtual register memory, allowing re-stateful applications to run simultaneously without interfering with switch operations (Zhu et al., 2022). Furthermore, Wang et al. used an isolation mechanism to improve packet processing performance. This mechanism allows each module to run securely and share resources. It is integrated with Menshen, which supports invariant processing behavior to maintain system performance (Wang et al., 2022).

3.2 *P4 on SDN/IP Network*

3.2.1 *In-Band Network Telemetry*

In-band Network Telemetry (INT) represents a novel approach to measuring network status, diverging from traditional measurement techniques and software-based measurements. INT employs a methodology whereby the metadata of each node is extracted and incorporated into packets, which are subsequently extracted at the final node. This approach enables the retrieval of status data in real-time, approaching real-time. However, inserting packets into the network introduces a significant packet overhead, potentially impacting network performance.

The development of an INT capable of monitoring network conditions in real-time and the implementation of a packet scheduling method that will be executed in the event of packet congestion to maintain network

performance was explained in the paper by J. Geng et al. (Geng et al., 2018). Furthermore, based on the research conducted by B. Guan et al., an INT can reduce overhead by imposing restrictions on monitoring nodes to optimize the monitoring process and reduce overhead (Guan & Shen, 2019). To Develop a flexible INT to classify packets using rate-based and event-based strategies, D. Suh et al. explained the method in their paper (Suh et al., 2020). This approach has the potential to reduce the overhead associated with INT packets.

The In-band Network Telemetry (INT) method proposed by PINT uses a probabilistic approach to set the maximum amount of overhead on each packet and divides the information into multiple packets. Three aggregation operations support encoding data accumulated into packets, namely per-packet aggregation, static per-stream aggregation, and dynamic per-stream aggregation (Ben Basat et al., 2020). Furthermore, a weight-based approach is used in an INT method designed to track and aggregate flows so that INT can run according to the network conditions (Mostafaei & Afridi, 2021). On the other hand, Osiński et al. implemented an end-to-end visibility method in INT that not only serves as a network condition meter but also as a network problem detector. This detection process is based on quality of service parameters (Osiński & Cascone, 2022).

3.2.2 *Security*

Security is a paramount concern in network operations. The current landscape of cyber-attacks is characterized by a high degree of diversity, with the potential to cause significant harm to users. Among the most prevalent forms of attack is the Distributed Denial of Service (DDoS) attack, which involves flooding a system or server with fake traffic until it becomes overwhelmed and fails. P4 provides a solution to detect and mitigate the data plane, thus enabling rapid handling.

Lapolli et al. proposed a real-time DDoS detection method in the data plane, utilizing Shannon entropy calculation as a parameter to distinguish fake and genuine traffic (Lapolli et al., 2019). On the other hand, Dimolianis et al. developed a DDoS detection scheme that utilizes various features, including the total amount of incoming traffic, network significance, and the symmetry ratio between incoming and outgoing packets. However, this method has the disadvantage of being unable to detect incoming packets below 5 Mpps (Dimolianis et al., 2020).

To reduce the errors of increasingly complex networks, Grewal et al. developed a system that utilizes Information Flow Control (IFC) methodology to protect confidentiality and integrity and offers flexibility, generality, and lightweight nature (Grewal et al., 2022). Kecskeméti et al. proposed a network security method that operates at a line rate, enabling rapid identification and mitigation of security threats. P4RROT implements various network security protocols, including those for NTP, ports, and packets (Kecskeméti et al., 2023). Reddy et al. introduced a security system that runs on the data plane by adding machine learning (ML) to detect attacks, using a decision tree algorithm to determine if an attack is occurring. The use of ML in packet classification can reduce attacks by 50% (Reddy et al., 2023). However, implementing ML necessitates the availability of extensive datasets to facilitate the recognition of attack types. Integrating network security on the data plane based on P4 programming offers advantages in the form of enhanced flexibility in managing packet processing and rapid response in detecting and resolving network threats.

3.2.3 *Caching*

Caching is a temporary storage mechanism for data. Deploying caching on a network can enhance network performance by storing previously requested content. When a second request for the same content is received, the request is not directed to the end-point server but instead to the nearest node or server that is caching the content. This reduces the latency associated with retrieving the requested content.

Zhao et al. introduced a data plane caching technique based on the least recently used (LRU) approach, potentially boosting performance by up to 35% (Zhao et al., 2023). Building on this, Friedman et al. designed a

more versatile caching framework that supports multiple strategies, including LRU, least frequently used (LFU), first-in, first-out (FIFO), and hyperbolic caching, allowing flexible configurations to meet different requirements (Friedman et al., 2023). In another approach, Sagkriotis et al. developed an in-network caching solution specifically to improve the performance and throughput of Kubernetes environments (Sagkriotis & Pezaros, 2022). While caching can improve network performance, the implementation of caching using P4 currently needs to be more effective for content that is large enough due to the limited memory capacity of P4.

3.2.4 *Load Balancing*

Load balancing is a technique employed to distribute traffic load across a network. It is necessary because a high load will inevitably reduce the network's overall performance. Ensuring a balanced load can more consistently maintain the network's performance.

Zhang et al. proposed the LBAS framework to implement load balancing in the data plane, utilizing a dynamic weight algorithm to reduce latency effectively (J. Zhang et al., 2020). Meanwhile, the study by Kulkarni et al. explored the implementation of various load-balancing techniques in the data plane using the P4 programming language, which showed promising results, particularly with the DPDK and SHELL techniques. However, despite the positive outcomes, these techniques still experienced performance fluctuations and lacked stability (Kulkarni et al., 2022). To address these challenges, Zheng et al. proposed a more adaptive approach by introducing a machine learning-based load balancer using the Reinforcement Learning (RL) method. The integration of RL allows the load balancer to learn from network behavior, creating a more responsive and efficient system (Zheng et al., 2023).

Furthermore, Barbette et al. emphasized the importance of ensuring connection consistency in load balancers through the concept of per-connection consistency (PCC), which is crucial for maintaining continuous server connections without compromising network performance. Their study successfully introduced a load balancer that supports PCC while still maintaining optimal performance (Barbette et al., 2022). Additionally, Xie et al. highlighted the current approach in load balancing, which uses a scheduling scheme based on ECMP and manages elephant and mouse flows within the network path. This approach often results in high loads and potential congestion. To address this issue, their research proposed a solution that predicts elephant flows and reschedules them to avoid congestion, thereby improving overall network performance (Xie et al., 2023).

3.2.5 *Congestion Control*

Congestion can occur when the network is busy or when many users are accessing it. In such cases, the network condition is congested, which decreases the network's performance. One method for overcoming congestion is Enhanced Congestion Notification (ECN), which is less effective than it could be. Shahzad et al. proposed Enhanced ECN on TCP using the ECN Intercept method, which marks ECN bits at a certain threshold to reduce additional traffic and notification time (Shahzad et al., 2020). On the other hand, Tukovic et al. introduced P4air to improve fairness in congestion control algorithms, where most of the current congestion control algorithms exhibit unfair behavior towards packets, which may ultimately degrade performance. P4air is proposed to optimize resource utilization and reduce RTT (Turkovic & Kuipers, 2020).

De Sensi et al. introduced the Canary method, a congestion reduction technique that uses a dynamic tree algorithm and incorporates the concept of load balancing. Data from each node is used as a parameter for congestion control (De Sensi et al., 2024). On the other hand, Wu et al. proposed P4SQA, a P4-based method on switches to guarantee quality of service (QoS) in software-defined networking (SDN). P4SQA uses two phases: classification and queue management. Classification is performed with the help of machine learning, and the method is able to improve the overall network performance (Wu et al., 2023).

3.3 *P4 on NDN Network*

3.3.1 *Forwarding*

In NDN networks, packets are sent by forwarding using forwarders. Signorello et al. implemented a Named Data Networking (NDN) network using the P4 programming language. This research explains the mechanism in detail, but there are still some things that could be improved, such as the limited number of prefixes, the absence of caching, and the use of an older version of the P4 language (Signorello et al., 2016). Miguel et al. addressed these shortcomings by increasing the number of prefixes, using the latest version of the P4 language, and introducing caching, an essential feature in NDN networks. With these measures, the performance of P4-based NDNs can be improved, as shown by previous studies. However, implementing the modifications to the switch library in native devices may present challenges (Miguel et al., 2018).

Karrakchou et al. proposed an enhanced architecture for Named Data Networking (NDN) networks, where, previously, each service was provided at the application layer. In this study, such services are moved to the network layer to meet the needs of the application layer, thereby improving performance and managing resources more effectively (Karrakchou et al., 2020). On the other hand, Takemasa et al. proposed a high-speed packet forwarding technique up to terabits by combining methods between switches and multiple servers to obtain large DRAMs used to store FIBs and prefixes. However, this research still needs to include a content store for caching (Takemasa et al., 2021).

Hou et al. developed an NFD forwarder using the P4 programming language, introducing a previously missing caching feature. This addition, achieved by leveraging the content store in NFD, enables faster response times by up to 60% (HOU et al., 2022). In contrast, Yu et al. proposed a separate caching mechanism, which decouples caching management from the forwarding plane. This separation helps the system maintain performance while offering improved speed compared to earlier approaches (Yu et al., 2022).

3.3.2 *Routing*

Guo et al. tried to incorporate the concept of Software-Defined Networking (SDN) into Named-Data Networking (NDN) to ease the integration of NDN networks. If this concept is successfully realized in a P4 environment, integration with other networks will become more accessible while proving that the NDN concept is useful. With this method, the number of routed packets can be reduced by 43%, and the routing convergence time is reduced by 12% (Guo et al., 2021).

3.3.3 *Security*

Liu et al. identified that adding cache to a P4-based Named Data Networking (NDN) network is still a challenge due to P4's small memory capacity and limited availability. In addition, content-centric security issues at the application layer must be tailored to the application, so P4 is used to add a security layer at the network level. Adding a separate cache for more expansive storage and a CPA-based encryption algorithm makes the P4-based NDN network more available for caching and has better security (Liu et al., 2022).

3.3.4 *Others*

Madureira et al. explain that the difference in networking concepts between Named Data Networking (NDN), which is content-based, and IP, which is address- or location-based, often causes them not to work well together. To address this, this research introduces NDN Fabric, a new architecture and protocol that integrates NDN networking for communication in the edge network and path-based communication in the core network. NDN Fabric utilizes P4 to detect packets and take action in determining routing instructions. This concept can increase speed up to 150 times and speed up network distribution to 100 times (Madureira et al., 2021).

Refaei et al. identified that tactical networks are typically characterized by Denied, Disrupted, Intermittently connected, and Limited-bandwidth (D-DIL) network environments and fluid and dynamic networks. In this research, traffic management uses P4 with two main components: a policy engine (to create

policies for applications with filters and actions) and an enforcement engine (to determine actions based on the set policies). This method can improve bandwidth efficiency, but to achieve better performance, a large enough policy database is required (Refaei et al., 2021).

Meanwhile, Zha et al. explained that NDN sends complex flow table entries, which results in many flow table entries, increases the interaction delay between the control and data planes, and reduces the speed. This research offers a solution using bitmask multicast with P4, which implements the Pending Interest Table (PIT) in the P4 register and changes the structure of the PIT; if there is an incoming request, the system will calculate the hash and find a suitable bitmask for multicast (Zha et al., 2022).

Rosa et al. address the challenges associated with managing the Forwarding Information Base (FIB) and Longest Name Prefix Matching (LNPM) in conventional Named Data Networking (NDN) architectures, particularly in terms of efficiency and latency. They introduce a novel approach that eliminates the reliance on hashing to enhance the speed of name matching and decrease latency in data retrieval. This method is implemented within a programmable data plane, enabling dynamic modifications to data management policies. Simulation results demonstrate that this approach can reduce name-matching latency by as much as 30% and enhance network throughput, thereby providing a more efficient solution for managing FIB and LNPM in NDN (Rosa & Silva, 2022).

Table 1. Summary of the P4 article and methods used

Group		Main Mechanism
Software-Defined Network	Compiler	Generates a target-agnostic switch code relying on a NetHAL (Vörös et al., 2018). Translate P4 program to PC program (Ibanez et al., 2019), Virtual Memory Register to the compiler (Zhu et al., 2022), Uses isolation mechanism and Menshen design (Wang et al., 2022).
	In-band Network Telemetry	Integrate SDN monitoring and Network Management Module using P4 (Geng et al., 2018), Using P4 to verify checksums and adding rules at ingress (Guan & Shen, 2019), Using Rate-based and Event-based strategies (Suh et al., 2020), Using telemetry information as a probabilistic value (Ben Basat et al., 2020), Using weighted-based technique and tracking flows (Mostafaei & Afridi, 2021), Uses eBPF to extent the Linux network stack (Osiński & Cascone, 2022).
	Security	Using Shannon entropy for DDOS characterization calculation (Lapolli et al., 2019), Combining multiple traffic features to detect DDOS attacks (Dimolianis et al., 2020), Using Information-flow control (IFC) system on the data plane (Grewal et al., 2022), Implement some security techniques on the data plane (Kecskeméti et al., 2023), Using machine learning to detect mitigate adversarial attacks on the data plane (Reddy et al., 2023).
	Caching	Caching is made on the data plane with the LRU method (Zhao et al., 2023), Adding keys to packets for caching detection (Friedman et al., 2023), Using CRAQ on P4 with PSA architecture (Sagkriotis & Pezaros, 2022).
	Load Balancing	Using LBAS framework by implementing partial dynamic weight algorithms (J. Zhang et al., 2020), Using short-lived flow and long-lived flow algorithms (Kulkarni et al., 2022), Using dynamic tree alforithm for congestion control (Zheng et al., 2023), Using neural computing and AQM algorithm (Barbette et al., 2022).
	Congestion Control	Using the ECN packet intercept method on end devices (Xie et al., 2023), Load balancing written in P4 language with multiple loads balancing strategies (Shahzad et al., 2020), Using the reinforcement learning method and implemented on a switch (Turkovic & Kuipers, 2020), Create a block to establish an arbitrary load balancing mechanism (De Sensi et al., 2024), Using Online Elephant Flow Prediction for load balancing (Wu et al., 2023).
Named-Data Networking	Forwarding	Implement NDN network with P4_14 (Signorello et al., 2016), Implement NDN network with P4_16 (Miguel et al., 2018), Implement services at the network layer (Karrakchou et al., 2020), Combination of switches and multiple servers for prefix storage (Takemasa et al., 2021),

Group		Main Mechanism
		Added cache by utilizing the cache from NFD (HOU et al., 2022), Added cache with the cache management separate from the forwarding plane (Yu et al., 2022).
	Routing	Integrating NDN and TCP/IP networks with SDN concept (Guo et al., 2021).
	Security	Encryption at the network layer with CPA-Based encryption algorithm (Liu et al., 2022).
	Others	Integrate SDN network (core network) with NDN (edge network) (Madureira et al., 2021), Add a policy to each application and determine the action according to the policy (Refaei et al., 2021), Change the PIT structure to maintain a fixed bit mask to multicast the corresponding interface (Zha et al., 2022), Utilizing a Hash-Free method for the FIB and LNPM (Rosa & Silva, 2022).

4. Challenge for Future Research

As illustrated in Table 1, many implementations and applications utilize the P4 language. Its adaptability and flexibility to technological developments is an added value. Nevertheless, there are still many challenges in P4 research, especially in future networks such as Named-Data Networking, which still needs to be widely published, so there are still many opportunities and challenges.

4.1 *Security*

Network security issues remain a concern as technological developments create diverse attack techniques. P4 can add security at the data plane level so that detection can be known earlier than application-level security (Garzón et al., 2024). These detection techniques are already quite good. Moreover, some already use ML, but in terms of mitigation, there are still some that are not good, and even there are still no mitigating actions, so it is necessary to improve mitigation with algorithms that are more efficient in choosing actions. In addition, with the ability of P4 to configure the data plane, security should be done without the intervention of the control plane so it can reduce its work time.

4.2 *Caching*

Caching is a network's most essential and valuable thing because it can improve its system performance. In IP networks, this caching system may or may not exist, but in NDN networks, caching is an essential component that must exist because it is one of the main components of the NDN (Content Store). However, due to the lack of memory available on the P4, finding a way to cache content on the NDN network is a challenge. Although there are already several ways, such as separating CS and modifying the P4 library for large memory, this will not be easy to implement. So, there is a need for a more efficient and better CS implementation algorithm with P4.

4.3 *In-Band Network Telemetry*

In-band network telemetry is a framework for monitoring networks by retrieving information from each switch node. Network monitoring with INT can provide benefits such as real-time data, flexibility, and low latency. INT has evolved extensively in IP networks and is used for monitoring and congestion control, security, load balancing, and others (Lü et al., 2023; Tan et al., 2021). However, in NDN networks, the implementation of INT still needs to exist, so it is a challenge for researchers because INT can provide many benefits to the network.

4.4 *Artificial intelligence*

Machine learning or deep learning is one of the current trends, and it is even used in a network. ML/DL makes the system smarter to provide benefits and improve the system's performance (Sapio et al., 2021). Therefore, using ML/DL in P4 is a challenge for researchers, as ML/DL can be used for network security, load balancing efficiency, congestion problems, and others.

4.5 *Adaptability and Integration*

Considering the existing network infrastructure, it is evident that adapting to new technologies can be challenging, especially when a complete overhaul is required. Even if significant performance improvements are achievable, such changes are unlikely to be implemented all at once. Therefore, integrating Programming Protocol-independent Packet Processors (P4) with the current infrastructure remains a challenge for the future. For instance, P4 can be utilized to enhance performance in 5G networks and Software Defined Networks (SDN) without necessitating a complete transformation of the underlying infrastructure (Alvarez-Horcajo et al., 2021; Memarian et al., 2024). Future work could involve integrating P4 with legacy switches, where P4 switches would function as monitoring entities that gather information from legacy switches and implement mitigation strategies in case of errors.

5. Conclusion

This paper maps existing research and discusses the Protocol for Programming Independent Packet Processor (P4) in detail, including its working mechanisms, components, advantages, and implementations. The implementations discussed are not only from Software-Defined Network (SDN)/IP networks but also cover their implementation in Named-Data Networking (NDN) networks. Implementations in SDN/IP networks are numerous and varied, such as Network Security, Load Balancing, Congestion Control, In-band Network Telemetry, etc. Conversely, in NDN networks, existing implementations are still limited, and current research focuses mainly on the content store.

P4 offers broad potential for use in various applications or systems. There are many examples of implementation in SDN/IP networks that demonstrate P4 flexibility and effectiveness. Additionally, P4 can be further developed within NDN networks in the future as it aligns well with the NDN forwarding concept. Thus, this study not only highlights current challenges but also identifies future research opportunities that can drive further innovation in P4 application deployment. Applications or systems built using P4 operate at the lower layer of network architecture; hence, they can shorten work time and enhance system efficiency.

References

Afanasyev, A., Burke, J., Refaei, T., Wang, L., Zhang, B., & Zhang, L. (2018). A Brief Introduction to Named Data Networking. MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM), 1–6. https://doi.org/10.1109/MILCOM.2018.8599682

Alvarez-Horcajo, J., Martínez-Yelmo, I., Lopez-Pajares, D., Carral, J. A., & Savi, M. (2021). A Hybrid SDN Switch Based on Standard P4 Code. IEEE Communications Letters, 25(5), 1482–1485. https://doi.org/10.1109/LCOMM.2021.3049570

Barbette, T., Wu, E., Kostic, D., Maguire, G. Q., Papadimitratos, P., & Chiesa, M. (2022). Cheetah: A High-Speed Programmable Load-Balancer Framework with Guaranteed Per-Connection-Consistency. IEEE/ACM Transactions on Networking, 30(1), 354–367. https://doi.org/10.1109/TNET.2021.3113370

Ben Basat, R., Ramanathan, S., Li, Y., Antichi, G., Yu, M., & Mitzenmacher, M. (2020). PINT: Probabilistic In-band Network Telemetry. Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, 662–680. https://doi.org/10.1145/3387514.3405894

Bosshart, P., Daly, D., Gibb, G., Izzard, M., Mckeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., & Walker, D. (2014). P4: Programming Protocol-Independent Packet Processors. ACM SIGCOMM Computer Communication Review, 44(3), 87–95.

Budiu, M., Research, V., & Dodd, C. (2017). The P4 16 Programming Language. ACM SIGOPS Operating Systems Review, 51(1), 5–14.

De Sensi, D., Costa Molero, E., Di Girolamo, S., Vanbever, L., & Hoefler, T. (2024). Canary: Congestion-aware in-network allreduce using dynamic trees. Future Gener. Comput. Syst., 152(C), 70–82. https://doi.org/10.1016/j.future.2023.10.010

Dimolianis, M., Pavlidis, A., & Maglaris, V. (2020). A Multi-Feature DDoS Detection Schema on P4 Network Hardware. 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 1–6. https://doi.org/10.1109/ICIN48450.2020.9059327

Friedman, R., Goaz, O., & Hovav, D. (2023). PKache: A Generic Framework for Data Plane Caching. Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, 1268–1276. https://doi.org/10.1145/3555776.3590826

Gartner, Inc. (2023). Hype Cycle for Emerging Technologies, 2023. https://www.gartner.com/doc/reprints?id=1-2I23LIF0&ct=240710&st=sb&utm_content=300102255&utm_medium=social&utm_source=linkedin&hss_channel=lcp-27118076

Garzón, C., Ríos-Guiral, S., Leal, E., Gutiérrez, S. A., & Botero, J. F. (2024). P4 Cybersecurity Solutions: Taxonomy and Open Challenges. IEEE Access, 12, 6376–6399. https://doi.org/10.1109/ACCESS.2023.3347332

Geng, J., Yan, J., Ren, Y., & Zhang, Y. (2018). Design and Implementation of Network Monitoring and Scheduling Architecture Based on P4. Proceedings of the 2nd International Conference on Computer Science and Application Engineering. https://doi.org/10.1145/3207677.3278059

Goswami, B., Kulkarni, M., & Paulose, J. (2023). A Survey on P4 Challenges in Software Defined Networks: P4 Programming. In IEEE Access (Vol. 11, pp. 54373–54387). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2023.3275756

Grewal, K., D'Antoni, L., & Hsu, J. (2022). P4BID: information flow control in p4. Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, 46–60. https://doi.org/10.1145/3519939.3523717

Guan, B., & Shen, S.-H. (2019). FlowSpy: An Efficient Network Monitoring Framework Using P4 in Software-Defined Networks. 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), 1–5. https://doi.org/10.1109/VTCFall.2019.8891487

Guo, X., Liu, N., Hou, X., Gao, S., & Zhou, H. (2021). An Efficient NDN Routing Mechanism Design in P4 Environment. 2021 2nd Information Communication Technologies Conference (ICTC), 28–33. https://doi.org/10.1109/ICTC51749.2021.9441639

HOU, S., HU, Y., TIAN, L., & DANG, Z. (2022). NFD.P4: NDN In-Networking Cache Implementation Scheme with P4. IEICE Transactions on Information and Systems, 105(4), 820–823. https://doi.org/10.1587/transinf.2021EDL8100

Ibanez, S., Brebner, G., McKeown, N., & Zilberman, N. (2019). The P4→NetFPGA workflow for line-rate packet processing. FPGA 2019 - Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 1–9. https://doi.org/10.1145/3289602.3293924

Karrakchou, O., Samaan, N., & Karmouch, A. (2020). ENDN: An Enhanced NDN Architecture with a P4-programmabIe Data Plane. Proceedings of the 7th ACM Conference on Information-Centric Networking, 1–11. https://doi.org/10.1145/3405656.3418720

Kaur, S., Kumar, K., & Aggarwal, N. (2021). A review on P4-Programmable data planes: Architecture, research efforts, and future directions. Computer Communications, 170, 109–129. https://doi.org/10.1016/j.comcom.2021.01.027

Kecskeméti, K., Györgyi, C., Vörös, P., & Laki, S. (2023). In-Network Security Applications with P4RROT. Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, 346–351. https://doi.org/10.1145/3565287.3617612

Kim, H., & Feamster, N. (2013). Improving network management with software defined networking. IEEE Communications Magazine, 51(2), 114–119. https://doi.org/10.1109/MCOM.2013.6461195

Kulkarni, M., Goswami, B., & Paulose, J. (2022). P4 based Load Balancing Strategies for Large Scale Software-Defined Networks. 2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), 1–7. https://doi.org/10.1109/ICAECT54875.2022.9807999

Lapolli, Â. C., Adilson Marques, J., & Gaspary, L. P. (2019). Offloading Real-time DDoS Attack Detection to Programmable Data Planes. 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 19–27.

Liu, N., Gao, S., Yu, L., & He, G. (2022). A Secure and Cached-Enabled NDN Forwarding Plane Based on Programmable Switches. Wireless Communications and Mobile Computing, 2022(1), 4466942. https://doi.org/https://doi.org/10.1155/2022/4466942

Lü, H. R., Li, Q., Shen, G. B., Zhou, J. E., Jiang, Y., Li, W. C., Liu, K., & Qi, Z. Y. (2023). Survey on In-band Network Telemetry. Ruan Jian Xue Bao/Journal of Software, 34(8), 3870–3890. https://doi.org/10.13328/j.cnki.jos.006635

Madureira, A. L. R., Araújo, F. R. C., Araújo, G. B., & Sampaio, L. N. (2021). NDN Fabric: Where the Software-Defined Networking Meets the Content-Centric Model. IEEE Transactions on Network and Service Management, 18(1), 374–387. https://doi.org/10.1109/TNSM.2020.3044038

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev., 38(2), 69–74. https://doi.org/10.1145/1355734.1355746

Memarian, M., Kassler, A., Grinnemo, K.-J., Laki, S., Pongracz, G., & Forsman, J. (2024). Utilizing Hybrid P4 Solutions to Enhance 5G gNB with Data Plane Programmability. 2024 15th IFIP Wireless and Mobile Networking Conference (WMNC), 47–54.

Miguel, R., Signorello, S., & Ramos, F. M. V. (2018). Named Data Networking with Programmable Switches. 2018 IEEE 26th International Conference on Network Protocols (ICNP), 400–405. https://doi.org/10.1109/ICNP.2018.00055

Mostafaei, H., & Afridi, S. (2021). P4Flow: Monitoring Traffic Flows With Programmable Networks. IEEE Communications Letters, 25(11), 3546–3550. https://doi.org/10.1109/LCOMM.2021.3109793

Osiński, T., & Cascone, C. (2022). Achieving End-to-End Network Visibility with Host-INT. Proceedings of the Symposium on Architectures for Networking and Communications Systems, 140–143. https://doi.org/10.1145/3493425.3502764

P4 16 Language Specification version 1.2.2. (2021). http://p4.org

Reddy, S. S., Nishoak, K., Shreya, J. L., Reddy, Y. V., & Venkanna, U. (2023). A P4-Based Adversarial Attack Mitigation on Machine Learning Models in Data Plane Devices. Journal of Network and Systems Management, 32(1), 5. https://doi.org/10.1007/s10922-023-09777-6

Refaei, T., Ha, S., Starr, R., & Steele, M. (2021). Using NDN and P4 for Effective Traffic Management in Tactical Networks. Proceedings - IEEE Military Communications Conference MILCOM, 2021-November, 577–582. https://doi.org/10.1109/MILCOM52596.2021.9653078

Rosa, E. C., & Silva, F. de O. (2022). A Hash-Free method for FIB and LNPM in ICN programmable data planes. 2022 International Conference on Information Networking (ICOIN), 186–191. https://doi.org/10.1109/ICOIN53446.2022.9687201

Sagkriotis, S., & Pezaros, D. (2022). Accelerating kubernetes with in-network caching. SIGCOMM 2022 Demos and Posters - Proceedings of the 2022 SIGCOMM 2022 Poster and Demo Sessions, Part of SIGCOMM 2022, 40–42. https://doi.org/10.1145/3546037.3546058

Sapio, A., Canini, M., Ho, C.-Y., Nelson, J., Kalnis, P., Kim, C., Krishnamurthy, A., Moshref, M., Ports, D., & Richtarik, P. (2021). Scaling Distributed Machine Learning with In-Network Aggregation. 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21), 785–808. https://www.usenix.org/conference/nsdi21/presentation/sapio

Shahzad, S., Jung, E. S., Chung, J., & Kettimuthu, R. (2020). Enhanced explicit congestion notification (EECN) in TCP with P4 programming. Proceedings - 2020 International Conference on Green and Human Information Technology, ICGHIT 2020, 35–40. https://doi.org/10.1109/ICGHIT49656.2020.00015

Signorello, S., State, R., François, J., & Festor, O. (2016). NDN.p4: Programming information-centric data-planes. 2016 IEEE NetSoft Conference and Workshops (NetSoft), 384–389. https://doi.org/10.1109/NETSOFT.2016.7502472

Suh, D., Jang, S., Han, S., Pack, S., & Wang, X. (2020). Flexible sampling-based in-band network telemetry in programmable data plane. ICT Express, 6(1), 62–65. https://doi.org/https://doi.org/10.1016/j.icte.2019.08.005

Takemasa, J., Koizumi, Y., & Hasegawa, T. (2021). Vision: toward 10 Tbps NDN forwarding with billion prefixes by programmable switches. Proceedings of the 8th ACM Conference on Information-Centric Networking, 13–19. https://doi.org/10.1145/3460417.3482973

Tan, L., Su, W., Zhang, W., Lv, J., Zhang, Z., Miao, J., Liu, X., & Li, N. (2021). In-band Network Telemetry: A Survey. Computer Networks, 186, 107763. https://doi.org/https://doi.org/10.1016/j.comnet.2020.107763

Turkovic, B., & Kuipers, F. (2020). P4air: Increasing Fairness among Competing Congestion Control Algorithms. 2020 IEEE 28th International Conference on Network Protocols (ICNP), 1–12. https://doi.org/10.1109/ICNP49622.2020.9259405

Vörös, P., Horpácsi, D., Kitlei, R., Leskó, D., Tejfel, M., & Laki, S. (2018). T4P4S: A Target-independent Compiler for Protocol-independent Packet Processors. 2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR), 1–8. https://doi.org/10.1109/HPSR.2018.8850752

Wang, T., Yang, X., Antichi, G., Sivaraman, A., & Panda, A. (2022). Isolation Mechanisms for High-Speed Packet-Processing Pipelines. 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), 1289–1305. https://www.usenix.org/conference/nsdi22/presentation/wang-tao

Wu, Q., Liu, Q., Jia, Z., Xin, N., & Chen, T. (2023). P4SQA: A P4 Switch-Based QoS Assurance Mechanism for SDN. IEEE Transactions on Network and Service Management, 20(4), 4875–4886. https://doi.org/10.1109/TNSM.2023.3280913

Xie, S., Hu, G., Xing, C., & Liu, Y. (2023). Online Elephant Flow Prediction for Load Balancing in Programmable Switch-Based DCN. IEEE Trans. on Netw. and Serv. Manag., 21(1), 745–758. https://doi.org/10.1109/TNSM.2023.3318752

Yu, L., Gao, S., Liu, N., Wang, H., & Su, W. (2022). A Cache-enabled NDN Forwarding Plane based on Programmable Switches. 2022 International Conference on Networking and Network Applications (NaNA), 152–156. https://doi.org/10.1109/NaNA56854.2022.00033

Zha, Y., Cui, P., Hu, Y., Lan, J., & Wang, Y. (2022). A Scalable Bitwise Multicast Technology in Named Data Networking. IEICE Transactions on Information and Systems, E105D(12), 2104–2111. https://doi.org/10.1587/transinf.2022EDP7057

Zhang, J., Wen, S., Zhang, J., Chai, H., Pan, T., Huang, T., Zhang, L., Liu, Y., & Yu, F. R. (2020). Fast Switch-Based Load Balancer Considering Application Server States. IEEE/ACM Transactions on Networking, 28(3), 1391–1404. https://doi.org/10.1109/TNET.2020.2981977

Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, kc, Crowley, P., Papadopoulos, C., Wang, L., & Zhang, B. (2014). Named data networking. SIGCOMM Comput. Commun. Rev., 44(3), 66–73. https://doi.org/10.1145/2656877.2656887

Zhao, Y., Liu, W., Dong, F., Yang, T., Li, Y., Yang, K., Liu, Z., Jia, Z., & Yang, Y. (2023). P4LRU: Towards An LRU Cache Entirely in Programmable Data Plane. Proceedings of the ACM SIGCOMM 2023 Conference, 967–980. https://doi.org/10.1145/3603269.3604813

Zheng, C., Rienecker, B., & Zilberman, N. (2023). QCMP: Load Balancing via In-Network Reinforcement Learning. Proceedings of the 2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing, 35–40. https://doi.org/10.1145/3607504.3609291

Zhu, H., Wang, T., Hong, Y., Ports, D. R. K., Sivaraman, A., & Jin, X. (2022). NetVRM: Virtual Register Memory for Programmable Networks. 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), 155–170. https://www.usenix.org/conference/nsdi22/presentation/zhu