The Evolution of edge caching architectures in IoT applications using Software-defined Networking (SDN)

Florian Weidner
Philipps-University Marburg, Germany
Department of Mathematics and Computer Science, Distributed Systems Group
February 09, 2025

***Abstract*—Internet of Things (IoT) applications are generating huge amounts of data. To be able to process the data efficiently, new strategies are needed. Caching data at the network's edge turned out to lower energy consumption, reduced network latency, and lower requests to the core network in the cloud. SDN is a technology that fits to manage the caching strategies and hides the complexity from the users. In this paper, we will summarize SDN, with its advantages and challenges, general applications, as well as applications in IoT. We will compare different edge caching strategies using SDN for IoT applications, showing the development of architectures over time. The first architecture is an early approach using SDN for cooperative caching in Internet Service Provider (ISP) networks, using the priority of the data to make the caching decisions. The second architecture also considers the lifetime of data in a transient IoT network. The third architecture uses the Moth-Flame Optimization (MFO) algorithm to optimize the caching strategy and clusters the network to get a better Quality of Experience (QoE).**

***Index Terms*—SDN, IoT, edge caching**

I. Introduction

With the development of the internet and the increasing complexity of networks, their management and configuration of them have become more complex and time-consuming. Technologies like mobile networks, cloud computing, multimedia applications and virtualization have a high need of bandwidth, high accessibility, and dynamic network configurations. These requirements are a challenge for traditional networks. [9] [12]

Traditional Networks are very hardware-centric. Routers and switches are used to manage the network traffic. The control plane is very tightly coupled with the data forwarding by the data plane. Since both are happening on the local device, the configuration and management of the network is very time-consuming. Software-defined Networking (SDN) addresses these issues. It decouples the control from the data plane and uses a centralized approach for managing the network devices. For that, a centralized software-based controller is used to manage the network devices over the control plane. This leads to easier configuration, more flexible forwarding, enhanced performance, and reduced costs. [9] [12]

There are different applications where SDN is used, like data centers, optical networks, or even small businesses. Also, for IoT services, using SDN turns out to be very beneficial.

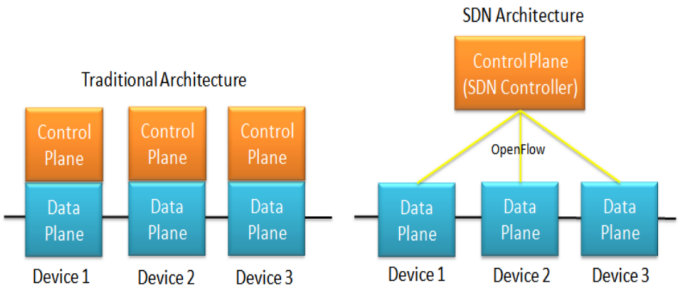


Fig. 1. Traditional Architecture and SDN Architecture [9]

IoT applications are often large-scale networks of heterogeneous devices, lacking flexibility, intelligence, and application-specific controls. SDN can help to overcome these problems by reducing management and adding flexibility to the network.

Edge caching is a popular strategy to optimize the usage of IoT network resources. It tries to store data on the devices distributed in the network. That reduces latency, energy consumption and reduces costly requests to the cloud. Also, the SDN Controllers can help to manage caching decisions based on the global view of the network and hide the complexity of the network from the end users. With edge caching, SDN nodes are used to cache data. Different architectures try to maximize the cache hit rate. The improvement of edge caching with SDN will be shown by comparing three different caching strategies.

In this paper, we will first summarize the concept of SDN and look at applications, advantages, and challenges. In section III, we will focus on the usage of SDN in IoT applications and which problems SDN can solve. In section IV, we deep dive into edge caching approaches using a SDN Controller in IoT applications. We look at three different architectures showing the development of edge caching with SDN. For each one, the architecture is described, the proposed improvements are explained, and the evaluation is presented. Finally, we will conclude our findings in section V.

II. SDN

"Software-defined Networking (SDN) is an emerging network architecture where network control is

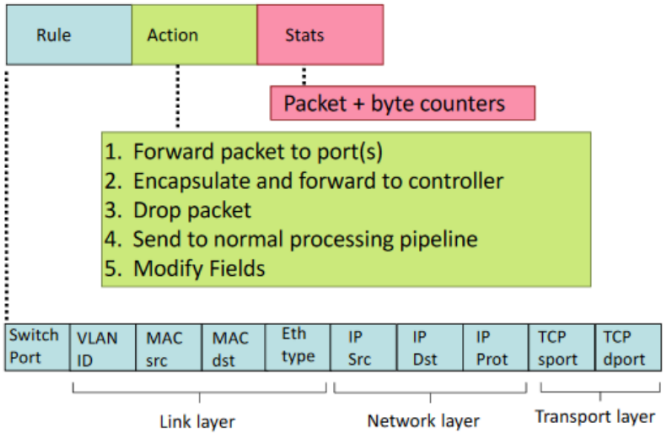


Fig. 2. Flow Table Entry Representation [13]

decoupled from forwarding and is directly programmable" [15]

This definition is by the Open Networking Foundation (ONF) from 2012. Software-defined Networking (SDN) decouples the control from the data plane. It uses a centralized controller with a global view of the network to manage the control plane. The controller can manage and adjust the network and the forwarding configuration in the flow tables of the network devices. There exist different implementations of SDN controllers. In figure 1, you can see the difference in the architectures of traditional networks and SDN networks. In the traditional architecture, each device has its own control plane to manage the device. With SDN, the control plane is centralized using the SDN controller.

The main responsibility of the data plane is forwarding the network traffic. For that, it uses flow tables to determine the forwarding destination, which are more complex forwarding tables of traditional routers or switches. More complex decisions based on the information of incoming packets are possible. In figure 2, you can see the representation of a flow table entry. An entry contains three columns. The first one contains the rules to match the incoming packages. The rules can be applied to any part of the datagram. The second one contains the actions that should be executed if the rule matches. The third column is used to store performance metrics on their corresponding rule and action fields. [13] The data plane can also be used to enable various functions like network inspection, anomaly detection, or traffic engineering. [12] The third plane is the application plane. On that plane, software is used to manage the network over the SDN controller. There, complex functions can be performed to configure or automate the network traffic, based on the customer needs.

Application Programming Interfaces (APIs) are used to communicate with the hardware in the network. The three planes use dedicated interfaces to communicate. The southbound interface is used to communicate between the control plane and the data plane. The northbound interface is used to communicate between the control plane and the application plane. The OpenFlow Protocol, maintained by the ONF is a commonly used open-source protocol defining an interface for the southbound communication between the controller and the network devices. It defines guidelines and uses Transmission Control Protocol (TCP) to update the flow table entries from the control plane. If the controller is distributed, the east-westbound APIs can be used to communicate between the controllers. [13]

A. Advantages and Challenges of SDN

Software-defined Networking (SDN) has several advantages compared to traditional networking. Here, some of them are summarized. SDN provides a better and easier network management. All network devices can be controlled from a single point. Also, newly added devices can be easily integrated into the network. [9] Additionally, the network's performance will be improved. It is possible to orchestrate the network traffic centrally. This leads to a better dynamic utilization of the network resources. This also leads to reduced costs. The management of the network is more efficient by using central software since there is less need to access the individual network devices directly. [9] The forwarding network devices can be simplified. They only need to be able to forward the network traffic and have basic functions to execute the controller's instructions, who takes over the management logic. [6] With SDN, the network gets programmable with applications that are installed on the control plane. The control plane can be directly programmed, since it is separated from the data plane. That also makes automation possible. [6]

However, SDN also faces some challenges. Research into SDN mainly focused on the control plane. The programmability of the data plane is not as advanced as the control plane. With OpenFlow, no solution is provided for data plane customization. [12] For forwarding, devices have a tradeoff in flexibility and performance. General purpose processors provide the highest flexibility, whereas specific standard products are specialized for high performance but lower flexibility. [9]New SDN switches use hardware combinations to achieve a better balance between flexibility and performance. [12] SDN networks depend on OpenFlow-compatible switches, limiting scalability. Also, the controller needs to be distributed to achieve further scalability over the limits of a single controller. Splitting the controller leads to typical distributed system problems like latency, fault tolerance, consistency and load balancing. On the other hand, it also leads to more resilience, performance, and availability. [12] Since SDN is widely being adopted and used, security is becoming very important. Controllers are a central target for security threads. With unauthorized access to the controller, the whole network can be compromised. Authentication between controllers and their network devices with Transport Layer Security (TLS) lightens these threads. An effective security model is mandatory to achieve secure network protection. [9]

B. Applications of SDN

SDN networks are used for data centers, enterprise networks, optical networks, and even homes and small businesses. SDN enables customization and deployment of new services or policies because of the independence of the control and the data plane. Therefore SDN can be used in various network environments. Data centers operate large-scale networks with high traffic, traffic management, and many policies. Here SDN can be used to manage the network traffic and to provide a better utilization of the network resources. Generally, the same works for enterprise networks. Also, for optical networks, the ONF provides specialized protocols to integrate multiple network technologies. And even for small networks, it turns out that using SDN is useful. Having a single point of control makes it easier to manage the network. [9]

III. Using SDN in IoT Applications

Internet of Things (IoT) connects devices with limited resources to create various services. IoT applications are created to mostly collect data and execute tasks for multiple domains, like industrial process systems, traffic monitoring, and a large variety of end-user applications. Often, they result in large-scale networks with many heterogeneous devices and protocols. [10] Li et al. identified that IoT faces the following problems:

- Difficulties in control and management due to the geographically distributed heterogeneous devices in various domains.
- Difficult to program and configure due to different devices with different capabilities, like memory constraints, bandwidth, and energy usage.
- Long service provisioning, due to the need for a complete development cycle, including installing, configuring, and testing the devices.
- Resources are not fully used. Devices have not been completely considered as network resources. [17]

Missing flexibility, intelligence and application-specific controls lead to these problems. SDN brings a global view of the network and provides capabilities to use network resources efficiently. It reduces the management and brings flexibility to mitigate the problems of IoT.

IoT devices need to be managed a lot due to the need for configurations, reconfigurations, resource allocations, and communication between the devices. [17] The concept of a central controller of a SDN network fits the need for central management for all devices in an IoT network. The controller can be used to activate and deactivate sensors based on the current needs. Also, the routing of sensor data can be optimized. [10] There exist multiple frameworks for managing IoT devices with SDN. [17] The integration is not trivial, since SDN mainly focuses on controlling traffic. It lacks the ability to control sensor hardware and IoT applications. [10]

Another application of SDN for IoT is for cellular networks. Multiple proposals exist for SDN-based cellular networks. With policies and a central controller, abstractions in geographical areas, load balancing, packet inspection, and packet processing can be achieved. [17]

The most common device in IoT are sensors. There are also proposals for SDN-based solutions for sensor networks. One example is the Software-Defined Wireless Sensor Network Framework (WSNSDN). It consists of a Base Station (BS) and several sensor nodes in a classical architecture. The BS is controlled by the SDN controller, managing the routing instead of the sensor nodes. The sensor nodes also contain flow tables like switches in the SDN architecture. [17] There also exist architectures using reconfiguration based on customer needs. That enables sharing a single infrastructure for multiple applications. Sensor OpenFLow (SOF) also propose reprogramming and retasking the sensor nodes with the control plane. [10] Applying SDN to low-power Wi-Fi networks, wireless sensors can achieve a lower power consumption because of low control traffic. [11]

SDN based IoT networks are also used for improved security, enabling authentication and authorization of the devices on the controller. With a global view over the network, the controller can handle connection approvals securely. It also helps run distributed firewalls or detect unauthorized malicious devices. [17] It limits the impact caused by security threads for most devices in the network. The controller on the other hand promotes to a central target for security threads. [10]

IoT and SDN are both topics that are widely researched and also used in the industry. The combination of them is still at an earlier stage. There are many proposals for standards, but no practical implementations are available. However, there is still much interest in proposing solutions for the different already-mentioned applications. [11]

The main challenge to efficiently use SDN networks for IoT applications is to use all network-wide information and knowledge to manage the network from the control plane. For that, data needs to be coordinated efficiently between the devices. Only that way, a collective intelligence can be achieved. Edge Computing tries to process data closer to the source rather than centralized in the cloud. It reduces latency and uses the existing resources in the network more efficiently. [2] Li et al. states different research approaches to use edge computing also in IoT applications using SDN.

IV. Caching in IoT with SDN

One part of edge computing, is caching data on the edges of the network. It is a strategy to optimize the usage of network resources, ensuring lower network latency, energy consumption, and higher availability. [7] [16] [8] Without caching, popular content will be transmitted repeatedly, including requests into the cloud, wasting network resources. Especially devices receiving the same information simultaneously. Often, the content needs to be location-sensitive, displaying different information on different locations of the world, and often, the cached contents are only used by devices in the same areas. To demonstrate the effect of caching on latency and energy consumption, we can look at the results of the caching strategy

proposed by [7]. In figure 9, you can see that the total energy consumption decreases with the cache size. With no caching, the energy consumption stays the same. When using caching, the energy consumption decreases significantly by increasing the cache size for all strategies. The same is shown in figure 10 for the average response latency. The constant latency with no caching is reduced by caching by more than half.

The network of a IoT application using SDN and caching at the edge typically consists of the following components:

- Edge SDN nodes: equipped with caching capabilities in order to serve multiple requests within the domain, without contacting the remote cloud [7] [16]
- SDN Controller: The central controller has the possibility to see all edge SDN nodes to manage their flow tables. It instructs the edge nodes to perform the caching actions. [7] [16]
- IoT gateway: used to transmit data between data plane devices, provides authentication and authorization, and ensures safety and security. [7]

Also, due to the huge amount of data collected in IoT networks, caching is needed to process the data locally before sending it to the cloud. Otherwise, useless data may be sent and stored centrally, producing unnecessary network traffic. [7] It also challenges the core network to allocate network resources for multiple data requests and replies from consumers. [16] A lot of different features are important for caching decisions:

- redundancy and data unavailability when the device is not available. [7]
- storage costs
- retrieval latency
- popularity of the data, which is the number of requests for the same data over a certain amount of time. [16]
- the lifetime of the data, which is the time the data is valid. [16]

The caching policies need to be dynamic. [3] found out that IoT data follows Zipf's law with a skewness parameter that can change regularly. The requested data from users of the IoT application is related to the users' needs, which is changing during the day. Chen et al. demonstrate that a period of 60 minutes is enough to capture a change in variation in the popularity.

The main goal of caching data at the edge is to improve the user experience. Users should not be aware of the complexity of the network. Caching decisions at the edge of a network are complicated, especially with heterogeneous devices. Here SDN can help to overcome these challenges with a centralized orchestration. A SDN controller can set policies for caching and hide the complexity of the heterogeneous network from the end users. Caching decisions will be decided based on the network-wide information of the controller, like the network topology or storage capabilities. When a user requests content produced by some sensor device nearby, the SDN controller can identify the location of the content cached and forward the request to the device, caching the requested data. The request

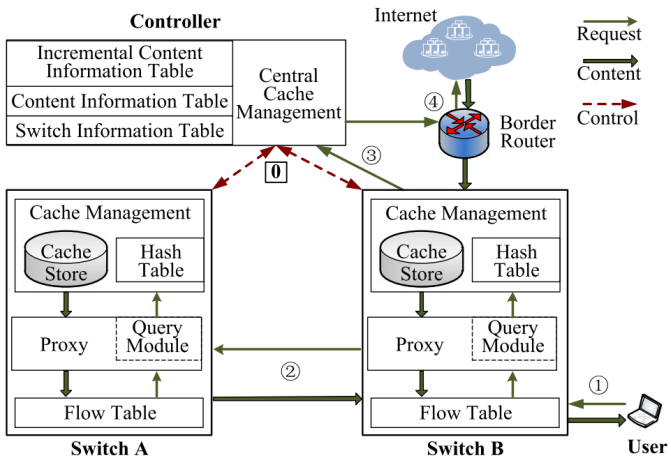


Fig. 1. The operation mechanism of SCCN.

Fig. 3. The operation mechanism of SCCN [4]

will be sent to the cloud if the data is not cached anywhere. [7] There are multiple applications where caching at edge nodes can be applied to improve the efficiency of the network:

- Smart health-care: Increased efficiency for processing in real-time with cached data
- Smart cities: Large amounts of data are generated, which must be processed nearby
- Internet of vehicles: For vehicle-to-vehicle communication, the collected data is only useful for other vehicles nearby.
- Intensive computation systems: Require low latency times.
- Wireless sensor networks [7]

Already existing caching strategies designed for traditional internet and multimedia content cannot be directly applied to IoT applications. Data like YouTube videos or movies will not change over time once available. They can be accessed even years after they are stored. Data stored in IoT networks are mostly transient. It expires after a certain amount of time. For example, in time series data, like pulse rates or pollution indicators, the lifetime of data states how long the data is reliable after being generated. After the expiration, the producer uploads it to the cloud. The heterogeneity of the edge devices also makes it hard to use already existing caching strategies. [16]

The following chapters will present different approaches for caching data in IoT networks with SDN at the edge.

A. Early approach using SDN for cooperative caching in ISP networks

In 2018 [4] proposes SDN-based Cooperative Cache Network (SCCN) for ISP networks using a SDN Controller. They propose a Relaxation Algorithm (RA) based on a relaxation-rounding technique [14] to solve the caching problem. They also provide a Heuristice Algorithm (HA) based on the concept of Circular Convex Set [5] to provide an efficient solution

usable for practical large-scale systems. The SCCN is implemented with Open vSwitch and Ryu controller.

> "[The] SCCN Controller can detect the most popular contents at the global level, perform intensive computations of content placement decisions and populate new request forward directions to SCCN Switches quickly" [4]

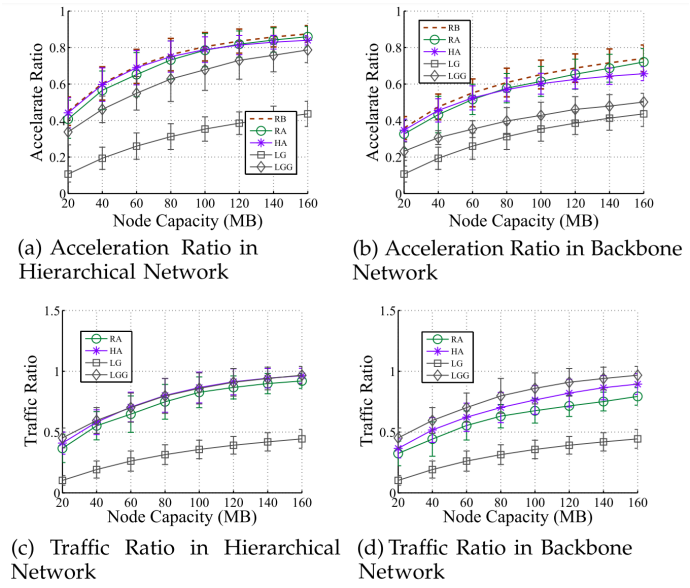
You can see the proposed architecture in figure 3. The Controller stores three tables of collected data from the network. The Switch Information Table (SIT) stores information about the switches containing cache capacity and bandwidth. The Content Information Table (CIT) tracks cached contents and calculated request counts and the Incremental Content Information Table (ICIT) stores information about recently requested data that was not cached yet. The Controller makes periodic content placement decisions based on the information in these tables. When the requested frequency of content exceeds a threshold within a time period, an additional caching decision is triggered. If a edge node receives a requeste and the data is not stored, it is forwarded to the Controller, which fetches the data and updates the flow table for future routing. The evaluation of SCCN compares the RA and HA algorithms with two algorithms proposed in [1]. The Local-Greedy Algorithm (LG) has a full replication strategy and each node caches the K most popular contents. Local-Greedy-Gen Algorithm (LGG) does not use a replication strategy and only stores a single copy of each content in the network. In figure 4, you can see the evaluation results with different cache sizes. You can see that the acceleration ratio (the ratio between the saved transmission delay and the original internet delay) and traffic ratio (the ratio between the saved traffic and desired traffic). The RA and HA algorithms outperform the LG and LGG algorithms in the acceleration ratio overall cache sizes. The RA and HA have very similar results, showing that the HA algorithm is a good approximation for the RA algorithm and should be used in large-scale systems. The RA and HA algorithm has a similar traffic ratio like the LGG, all outperforming the LG. The evaluation also states that the average internet delay, impact of request patterns, and large problem sizes do not really affect the better performance of SCCN.

B. Using the lifetime for a caching decision

In 2021 [16] proposes a caching policy also using a SDN-controller.

> "Such a policy identifies the *IoT contents* to be cached and *the cachers*, by *jointly* accounting for the *IoT content popularity and lifetime*" [16]

Their goal was to prioritize the cache hit ratio for the most popular contents, which have a longer lifetime expectancy. The policy should reduce the retrieval latency and maximize the content diversity available at the edge domain even more. Using the OpenFlow protocol for the southbound interface, the policy should run as a network application on top of the SDN Controller. They define the optimal content placement



(a) Acceleration Ratio in Hierarchical Network

(b) Acceleration Ratio in Backbone Network

(c) Traffic Ratio in Hierarchical Network

(d) Traffic Ratio in Backbone Network

Fig. 4. Acceleration ratio and traffic ratio with different cache size [4]

through an Integer Linear Programming (ILP) problem that includes their named objectives. They claim to be the first caching optimization approach in a distributed edge domain that considers the popularity and lifetime of IoT data. The idea is that caching very popular content with only a very short lifetime left is less valuable than less popular content with a longer lifetime. Over a longer lifetime, the data may have more requests than the popular one. The SDN Controller instructs the caching actions at regular intervals. Due to the change of popularity over time [3] they decided to set the interval at 60 minutes. The SDN nodes then store the data until the data lifetime expires or a new caching decision is made. The SDN controller monitors the topology and measures the packet loss probability and the link delay to estimate the content retrieval latency. Using the OpenFlow protocol, it also monitors the capabilities and content statistics of the SDN nodes. That gives information about the amount of flow table entries or buffer sizes of the switches. Also, the request arrival rates are summed up by giving the Controller information about all content requests. This information is used to make the caching decisions. If a node needs to get specific data, it sends a request to the Controller, which injects the forwarding rules into the flow table of the SDN node, so that it can retrieve the data from the cacher directly. To maximize the content diversity, they store only one content copy in the edge network, similar like the LGG algorithm of [1]. That reduces traffic transit costs because more space is available for distinct data, and therefore, less traffic has to leave the domain requesting data at the cloud. The ability of the SDN Controller to use load balancing helps against intra-domain congestion problems.

For the optimal content placement, they formulate the problem as an ILP problem. It is a NP-hard problem, which makes it computationally intensive and impractical for large-scale applications. Therefore, they propose a heuristic algorithm

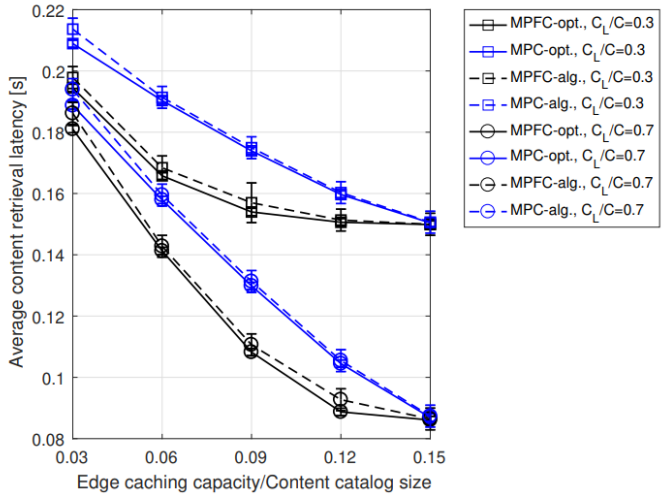


Fig. 5. Average content retrieval latency [16]

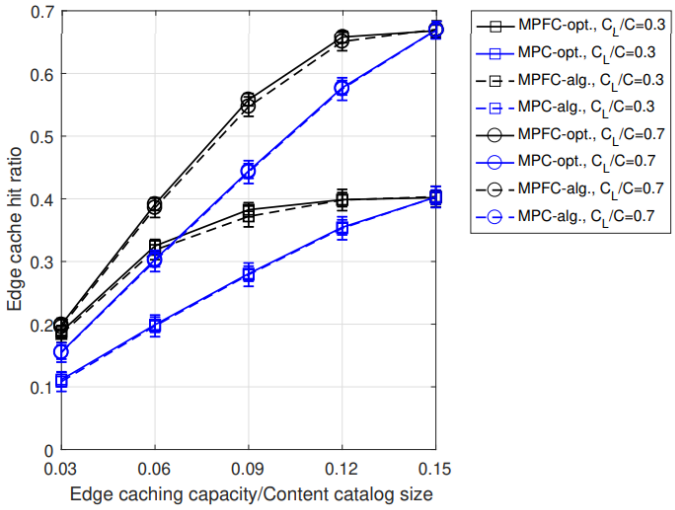


Fig. 6. Average content retrieval latency [16]

using a greedy approach to find near-optimal solutions more efficiently. This heuristic evaluates factors such as node storage capacity, network link bandwidth, and the transient nature and popularity of IoT content to make caching decisions.

The evaluation of their caching policy shows that it outperforms other caching strategies in terms of cache hit ratio, content diversity, and retrieval latency. They tested their solution in three scenarios to the benchmark of Most Popular Contents (MPC) which is an approach based on the discussed algorithm of [1] in section IV-A. In all three scenarios, Most Popular Fresh Contents (MPFC) achieved lower content retrieval latency, higher cache hit rates, and higher freshness probabilities. In figures 5, 6 and 7, you can see the results of the first scenario. You can see that the content retrieval latency and the cache hit rate from MPFC(black lines) is better on all cache sizes than MPC(blue lines). For the freshness probability, with increasing cache size, the freshness probability of MPFC equals MPC. The paper proves that the usage of the content's lifetime is improving the caching decisions in IoT networks.

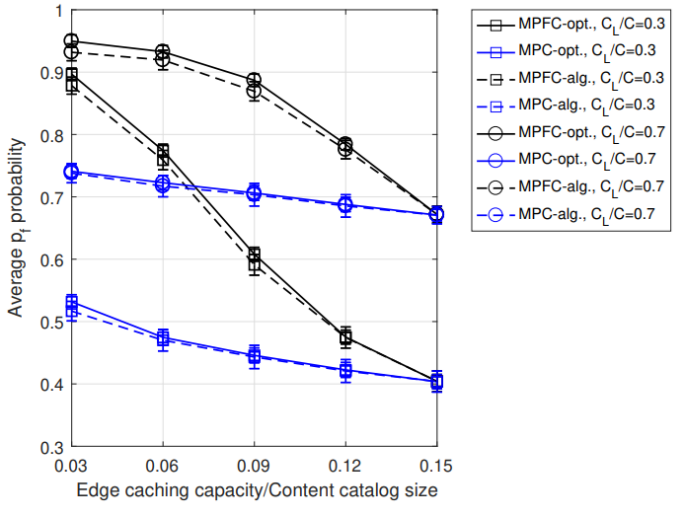


Fig. 7. Average content retrieval latency [16]

C. Moth-flame optimization algorithm

In [7] from 2023, the authors propose a caching strategy using the Moth-Flame Optimization (MFO) and cluster the network to get a better Quality of Experience (QoE). It is a population-based metaheuristic inspired by the navigation behavior of moths. Light sources cause moths to fly in a straight line toward them and fly spirally around a light source when very close. They use the algorithm to group similar content and determine the optimal locations for caching specific data. They cluster the edge nodes and select cluster heads using their proposed algorithm Moth-Flame Optimization-Cluster Head Selection (MFO-CHS).

In their architecture, they also use SDN with a centralized SDN Controller. It uses the OpenFlow protocol for southbound communication, and RESTful APIs for the northbound interface. The network architecture is displayed in figure 8. You can see three layers. In the lowest layer are the IoT devices, like sensors or actuators, clustered. Each one is connected to a SDN edge node on the edge computing layer. Here, the caches are installed. The nodes are connected to the SDN Controller, seeing an overview of the network. The locally distributed SDN Controllers are connected to the global SDN Controller being in the cloud computing layer. It coordinates the physical distribution of the SDN Controllers and communication to the cloud.

For the caching decisions, they use two policies, one for decision-making and one for caching replacement. The first one decides with the Moth-Flame Optimization-Edge Caching (MFO-EC) algorithm on a time interval what data should be cached where. The replacement policy states which data should be deleted if the cache, the decision-making policy selected, is full. The SDN Controller uses topology information and IoT data to make caching decisions. The algorithm also uses content freshness and content popularity as features

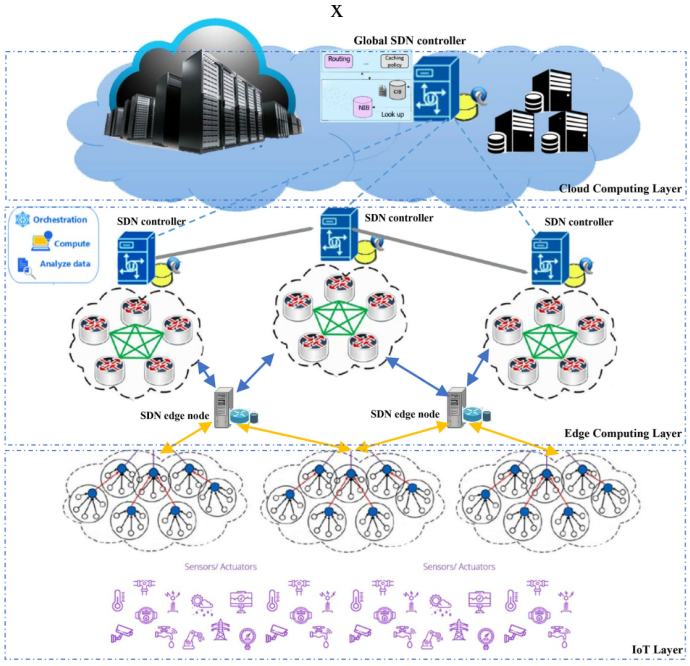


Fig. 8. The structure of the proposed model [7]

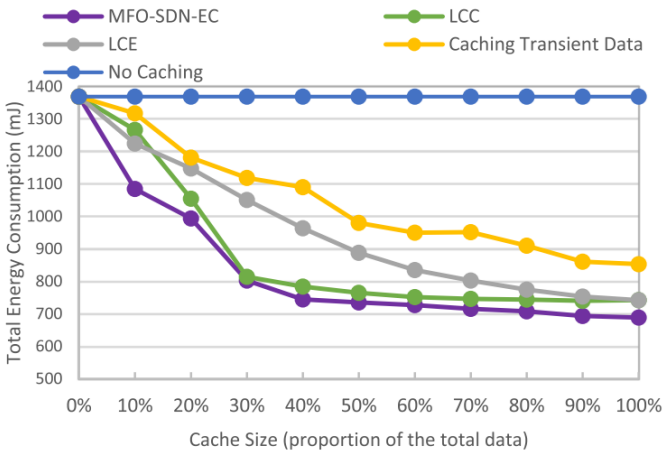


Fig. 9. Total Energy Consumption (mJ) VS cache size [7]

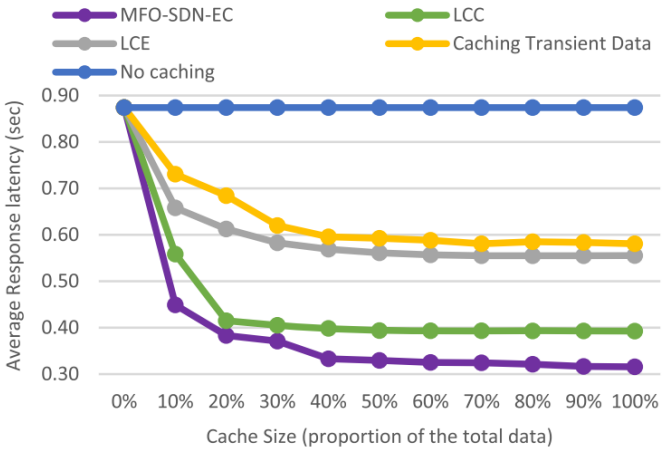


Fig. 10. Average Response latency (sec) VS cache size [7]

to make caching decisions. From these two features, caching weights are calculated, which are then used by the MFO-EC algorithm.

To create an efficient environment to be able to process massive amounts of data, the authors propose to use IoT clusters. It further reduces the latency and energy consumption in the network, which leads to better QoE for the users. Edge networks are created to be able to better use and share their computational power and resources. This complexity is hidden to the users with the SDN Controller. The MFO-CHS algorithm is used to select the cluster heads dynamically. Selecting the best cluster head brings lower end-to-end latency and packet drops. The algorithm tries to select the ”fittest” node in the cluster and ensure that the network’s energy consumption is at the minimum. For the fitness function, the distance between nodes, the residual energy of the node, and the priority assigned to the node are considered by the MFO-CHS algorithm. Nodes get priority if they need lower latency or transfer critical data. The clusters are changing over time, adapting to the state of the network. After simulating, the authors found out that their proposed caching strategy outperforms other caching strategies regarding energy consumption, cache hit rate, and average response latency. You can see the results in figures 9, 10 and 11. They compare their architecture to different caching strategies. The best architecture the MFO architecture is compared to is a lifetime-based architecture called Lifetime-based Cooperative Caching (LCC) from [18], which proposes a caching strategy based on the lifetime and popularity of the data but without using SDN like the presented strategy [16] in section IV-B. It outperforms the LCC strategy in all three metrics by a bit. The evaluation shows that using the moth-flame algorithm and clustering the network improves the network efficiency.

V. Conclusion

After introducing the topic, we summarized the concept of SDN and looked at applications, its advantages, and challenges. Then, we focused on the usage of SDN in IoT applications and why it helps to overcome problems of IoT applications on a large scale. We focused on using edge caching managed by a SDN controller to improve the network efficiency. We presented three papers showing the evolution of caching strategies in IoT networks with SDN. The first paper [4] proposed a caching strategy for ISP networks using the contents priority to make caching decisions for the edge nodes. The second paper [16] extended the caching policy by using the priority and the lifetime of the data to make caching decisions. The third paper [7] proposed a caching strategy, taking priority and lifetime of the data into account, using the MFO algorithm and clustering the network to improve the efficiency even more.

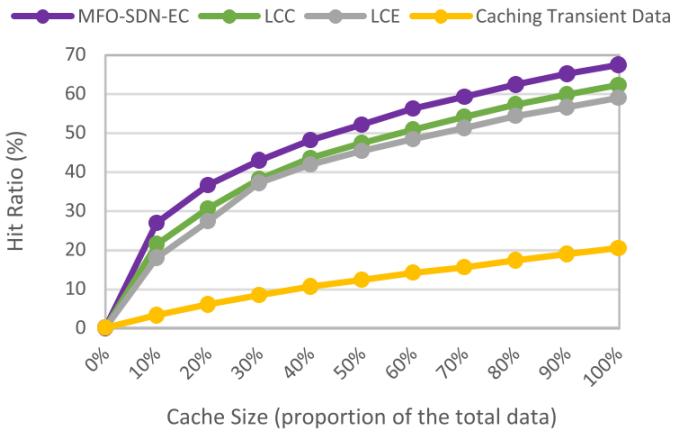


Fig. 11. Hit Ratio VS cache size [7]

REFERENCES

[1] Sem Borst, Varun Gupta, and Anwar Walid. "Distributed Caching Algorithms for Content Distribution Networks". In: *2010 Proceedings IEEE INFOCOM*. 2010, pp. 1–9. DOI: 10.1109/INFCOM.2010.5461964.

[2] Keyan Cao et al. "An Overview on Edge Computing Research". In: *IEEE Access* 8 (2020), pp. 85714–85728. DOI: 10.1109/ACCESS.2020.2991734.

[3] Bo Chen et al. "IoTCache: Toward Data-Driven Network Caching for Internet of Things". In: *IEEE Internet of Things Journal* 6.6 (2019), pp. 10064–10076. DOI: 10.1109/JIOT.2019.2935442.

[4] Yong Cui et al. "SDN-Based Big Data Caching in ISP Networks". In: *IEEE Transactions on Big Data* 4.3 (2018), pp. 356–367. DOI: 10.1109/TBDATA.2017.2651901.

[5] Dorit S. Hochbaum. "Heuristics for the fixed cost median problem". In: *Math. Program.* 22.1 (Dec. 1982), 148–162. ISSN: 0025-5610. DOI: 10.1007/BF01581035. URL: https://doi.org/10.1007/BF01581035.

[6] Mudassar Hussain et al. "Software-defined networking: Categories, analysis, and future directions". en. In: *Sensors (Basel)* 22.15 (2022), p. 5551.

[7] Seyedeh Shabnam Jazaeri et al. "An efficient edge caching approach for SDN-based IoT environments utilizing the moth flame clustering algorithm". In: *Cluster Computing* 27.2 (May 2023), 1503–1525. ISSN: 1386-7857. DOI: 10.1007/s10586-023-04023-9. URL: https://doi.org/10.1007/s10586-023-04023-9.

[8] Seyedeh Shabnam Jazaeri et al. "Composition of caching and classification in edge computing based on quality optimization for SDN-based IoT healthcare solutions". en. In: *J. Supercomput.* 79.15 (2023), pp. 17619–17669.

[9] Abigail Jefia, S Popoola, and Atayero. "Software-defined networking : Current trends , challenges , and future directions". en. In: *Popoola, S. (2018, September). Software-Defined Networking: Current Trends, Challenges3rd North American International Conference on Industrial Engineering and Operations Management* (2018).

[10] Yuhong Li et al. "Enhancing the Internet of Things with knowledge-driven Software-defined Networking technology: Future perspectives". en. In: *Sensors (Basel)* 20.12 (2020), p. 3459.

[11] Kamaran H Manguri and Saman M Omer. "SDN for IoT environment: A survey and research challenges". en. In: *ITM Web Conf.* 42 (2022), p. 01005.

[12] Rahim Masoudi and Ali Ghaffari. "Software defined networks: A survey". In: *Journal of Network and Computer Applications* 67 (2016), pp. 1–25. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2016.03.016. URL: https://www.sciencedirect.com/science/article/pii/S1084804516300297.

[13] Alexander Nunez et al. *A Brief Overview of Software-Defined Networking*. 2023. arXiv: 2302.00165 [cs.NI]. URL: https://arxiv.org/abs/2302.00165.

[14] Zeev Nutov, Israel Beniaminy, and Raphael Yuster. "A (1–1/e)-approximation algorithm for the generalized assignment problem". In: *Operations Research Letters* 34.3 (2006), pp. 283–288. ISSN: 0167-6377. DOI: https://doi.org/10.1016/j.orl.2005.05.006. URL: https://www.sciencedirect.com/science/article/pii/S0167637705000611.

[15] Open Networking Foundation. *Software-Defined Networking: The New Norm for Networks*. White Paper. Open Networking Foundation, Apr. 2012. URL: https://opennetworking.org/wp-content/uploads/2011/09/wp-sdn-newnorm.pdf.

[16] Giuseppe Ruggeri et al. "Caching popular transient IoT contents in an SDN-based edge infrastructure". In: *IEEE Trans. Netw. Serv. Manag.* 18.3 (2021), pp. 3432–3447.

[17] Sahrish Khan Tayyaba et al. "Software Defined Network (SDN) Based Internet of Things (IoT): A Road Ahead". In: *Proceedings of the International Conference on Future Networks and Distributed Systems*. ICFNDS '17. Cambridge, United Kingdom: Association for Computing Machinery, 2017. ISBN: 9781450348447. DOI: 10.1145/3102304.3102319. URL: https://doi.org/10.1145/3102304.3102319.

[18] Zhe Zhang et al. "IoT Data Lifetime-Based Cooperative Caching Scheme for ICN-IoT Networks". In: *2018 IEEE International Conference on Communications (ICC)*. 2018, pp. 1–7. DOI: 10.1109/ICC.2018.8422100.