Comparing caching approaches with Software-defined Networking (SDN) for Internet of Things (IoT) applications

Florian Weidner
Philipps-University Marburg, Germany
Department of Mathematics and Computer Science, Distributed Systems Group
February 09, 2024

***Abstract*—The abstract goes here.**
***Index Terms*—SDN, IoT, caching**

I. Introduction

With the development of the internet and the increasing complexity of networks, the management and configuration of them become more complex and time consuming. Technologies like moblie 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.

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 form the data plane and uses a centralized approach for managing the network devices. For that a centralized software-based controller is used managing the network devices over the controll plane. This leads to easier configuration, more flexible forwarding, enhanced performance and reduced costs. [6] [9] There are different applicaions where SDN is used, like data centers, optical networks or even small businesses. Also for IoT services, using SDN turned out to be very beneficial. IoT applications often are large-scale networks of heterogeneous devices, with missing flexibility, intelligence and application-specific controls. SDN can help to overcome these problems, by reducing management and adding flexibility to the network.

A very popular strategy to optimize the usage of IoT network resources is edge caching. Also there SDN Controllers can help to manage caching decision based on the global view of the network and hiding the complexity of the network from the end users. With egde caching, SDN nodes are used to cache data to reduce latency and energy consumption of the network. Different architectures try to miximize the cache hitrate. The imporvement of edge caching with SDN will be shown, by comparing three different caching strategies.

In this paper we will first summerize the concept of SDN and look at applications and challanges In Section III we will focus on the usage of SDN in IoT applications. In Section IV we then deep dive even more into caching approaches with SDN for IoT applications. Multible caching strategies will be compared. Finally, we will conclude our findings in Section V.

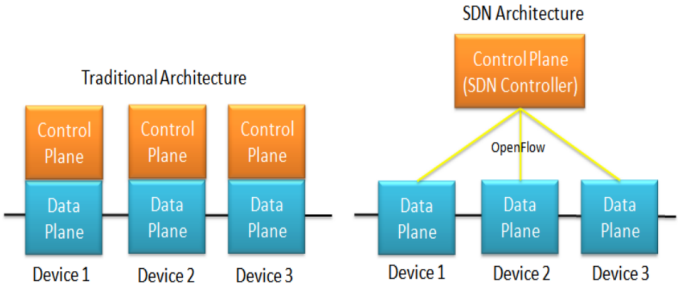

Fig. 1. Traditional Architecture and SDN Architecture [6]

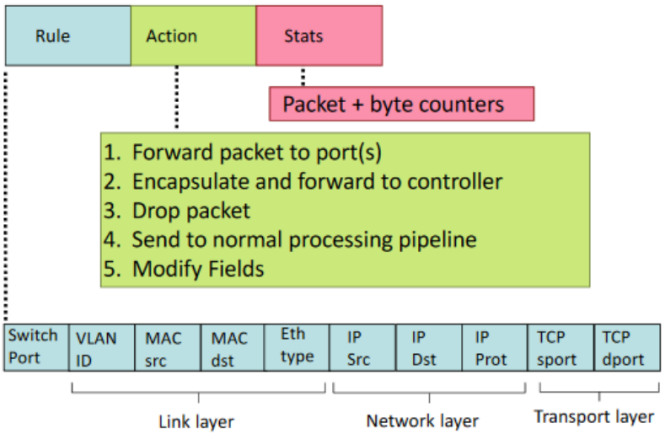

Fig. 2. Flow Table Entry Representation [10]

II. SDN

> "Software-defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding and is directly programmable" [11]

This definition is by the Open Networking Foundation (ONF) from 2012. Software-defined Networking (SDN) de-

couples the control form the data plane. It uses a centralized controller, which has a global view to the network to manage the control plane. The controller can manage and adjust the network and the forwarding configuration of the network devices. There exist different implementations of SDN controllers. In II 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. In SDN the control plane is centralized using the SDN controller.

The main responsitblity of the data plane is forwarding the network traffic. For that, it uses flow tables to determin the forwarding desitnation, which are more complex forwarding tables of traditional routers or switches. More complex decisions based on the information of incoming packtes are possible. In II you can se the representation of a flow table entry. An Entry contians 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 are 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 field. [10] The dataplane can also be used, to enable various functions like network inspection, anomaly detection or traffic engineering. [9] The third plane is the application plane. On that plane software is used to manage the network over the SDN controller. There complex funtions can be performed to configure or automate the network trafic, based on the customer needs. Application Programming Interfaces (APIs) are used to comunicate with the hardware in the network.

The three planes use dedicated interfaces to comunicate. The southbound interface is used to comunicate between the control plane and the data plane. The northbound interface is used to comunicate 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 entires from the control plane. If the controller is distributed the east-westbound APIs can be used to comunicate between the controllers. [10]

A. Advantages and Challenges of SDN

Software-defined Networking (SDN) has compared to traditional networking several advantages. Here the some of them are summerized.SDN provides a better and easier management of the network. All network devices can be controlled from a single point. Also newlly added devices can be easily integrated into the network. [6] Also the performance of the network will be imporved. It is possible to orchistrate 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 a central software, since there is less need to acces the individual network devices directly. [6] The forwarding network devices can be simplified. They only need to be able to forward the network traffic and have basic functions to be able to execute the instructions of the contorller, which takes over the management logic. [3] With SDN the network gets programmable with applicaitons that are installed ot the control plane. The control plane can be dircetly programmed, since it is seperated from the data plane. That also makes automation possible. [3]

But SDN also faces some challenges. Research into SDN mainly focused on the control plane. The programability of the dataplane is not as advanced as the control plane. With OpenFlow, there is no solution provided for data plane customization. [9] For forwarding devices have a tradoff in flexibilty and performance. General purpose processors provide the highest flexibilty wheras specific standard products are specialized for high performance but lower flexibilty. [6]New SDN switches are using hardware combinations to achive a better balance between flexibilty and performance. [9] SDN networks are dependent on OpenFlow compatible switches, which limits the scalability. Also the controller needs to be distributed to achive further scalabiltiy, 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 and it also leads to more resilicance, performance and availability. [9] Since SDN is widely being adopted and used, security is getting 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) lighten these threads. To achive a secure network protection, an effective security model is mandatory. [6]

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. [6]

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. [7] 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 full development cycle, including installing, configuring and testing the devices.
- Resources are not fully used. Devices haven't been completely considered as network resources. [14]

Missing flexibility, intelligence and application-specific controls, lead to these problems. SDN brings a global view on 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. [14] 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. [7] There exist multiple frameworks for managing IoT devices with SDN. [14] The integration is not trivial, since SDN mainly focuses on controlling traffic. It lacks the ability to control sensor hardware and IoT applications. [7]

Another application of SDN for IoT is for cellular networks. There are multiple proposals 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. [14]

The most common device in IoT are sensors. For sensor networks, there are also proposals for SDN-based solutions. 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. [14] 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. [7] Applying SDN to low-power Wi-Fi networks, wireless sensor can achieve a lower power consumption because of low control traffic. [8]

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, approvals of connections can be handled securely by the controller. It also helps run distributed firewalls or detect unauthorized malicious devices. [14] It limits the impact caused by security threads for the most devices in the network. The controller on the other hand promotes to a central target for security threads. [7]

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. Yet, there is still a lot of interest proposing solutions for the different applications already mentioned. [8]

The main challenge to efficiently use SDN networks for IoT applications is to use all network-wide inforamtion and knowledge to manage the network form 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 that centralized in the cloud. It reduces latency and uses the existing ressouces in the network more efficiently. [1] Li et al. states different research approachs to use that concept also in IoT applications using SDN.

[13] [14]

IV. Caching in IoT with SDN

Next to edge computing, caching data on the edges of the network is another strategy to optimize the usage of network resources, ensuring lower network latency, energy consumption and higher availability. [4] [12] [5] Without caching, popular content will be transmitted repeatedly, 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 the devices in the same areas. To demonstrate the effect of caching for latency and energy consumption we can look at the results of the caching strategy proposed by [4]. In figure IV-B you can see that the total energy consumption decreases with the cache size. With no caching the energy consumption stays the same. If you use caching, for all strategies the energy consumption decreases significantly by increasing the cache size. The same is shown in figure IV-B 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 typicaly consists of the following components:

- Edge SDN nodes: equiped with caching capabilities in order to serve multiple requests within the domian, without contacting the remote cloud [4] [12]
- 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. [4] [12]
- IoT gateway: used to transmit data between data plane devices, provides authentication and authorization and ensures safety and security. [4]

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 get sent and stored centrally and unnecessary network traffic is

produced. [4] It also challanges the core network to allocate network resources for multiple data requests and replies from consumers. [12] A lot of different features are important for caching decisions:

- redundancy and data unavailability when the device is not available. [4]
- storage costs
- retreival latency
- popularity of the data, which is the number of requests for the same data over a certain amount of time. [12]
- the lifetime of the data, which is the time the data is valid. [12]

The caching policies needs to be dynamic. [2] found out that IoT data follows the Zipf's law with a skewness parameter that can change on a regular basis. The requested data from users of the IoT applicaion is related to the needs of the users, 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 challanges 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 makes a request for content produced by some sensor device nearby, the SDN controller can identify the location of the content cached and foreward the request to the device, caching the requested data. If the data isn't cached anywhere, the request will be sent to the cloud. [4] 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 genereated, 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 [4]

Already existing caching strategies that are designed for traditiondal internet and multimedia contents cannot be directly applied to IoT applications. Data like Youtube videos or movies won't change over time once available. They can be accessed even years after they are stored. Data stored in IoT networks are mostly transient, they expire after a certain amount of time. For example timeseries data, like pulse rates or indicators of pollution, 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 heterogenity of the edge devices also makes it hard to use already existing caching strategies. [12]

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

A. Using the lifetime for a caching decision

In 2021 [12] 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*" [12]

Their goal was to prioritze the cache hit ration for the most popular contents which have a longer lifetime expectancy. The policy should reduce the retreival latency and maximaize the content diversity available at the edge domain. The policy should run as a network application on top of the SDN Controller using the OpenFlow protocol for the southbound interface. They define the optimal content placement through an Integer Linear Programming (ILP) problem, that include their named objectives. They claim to be the first caching optimization aproach in a distributed edge domain, that are taking the popularity and the lifetime of IoT data into account. The idea is, that caching very popular content with only a very short lifetime left is less usefull than less popular content with a longer lifetime. Over the longer livetime it may have more total requests that the popular one. The SDN Controller instructs the caching actions at regular intervals. Due to the change of popularity over time [2] they decided to set the intervall at 60 minutes. The SDN nodes are then storing the data unitl 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 retreival 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 is summed up, by giving the Controller information about all content requests. These informations are used to make the caching decisions. If a node needs to get certain 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 retreive the data from the cacher directly. To maximize the content diversity, they store only one content copy in the edge network. That reduces traffic transit costs, because more space is available for distinct data and therefore less traffic has to leave the domain requesing 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 using a greedy approach to find near-optimal solutions more efficiently. This heuristic evaluates factors such as node storage capacity, network link

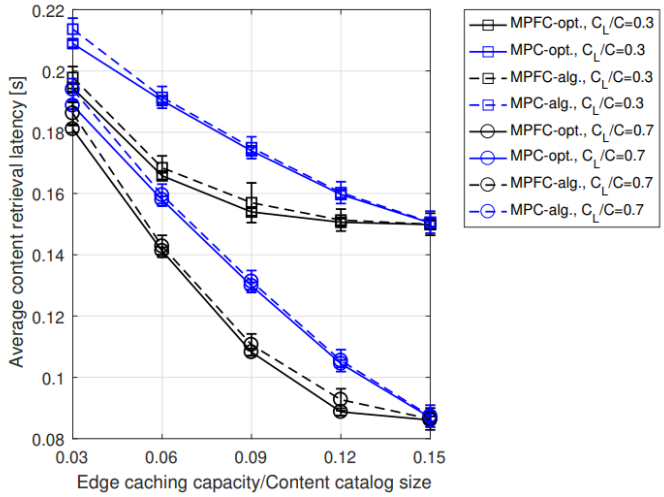

Fig. 3. Average content retrieval latency [12]

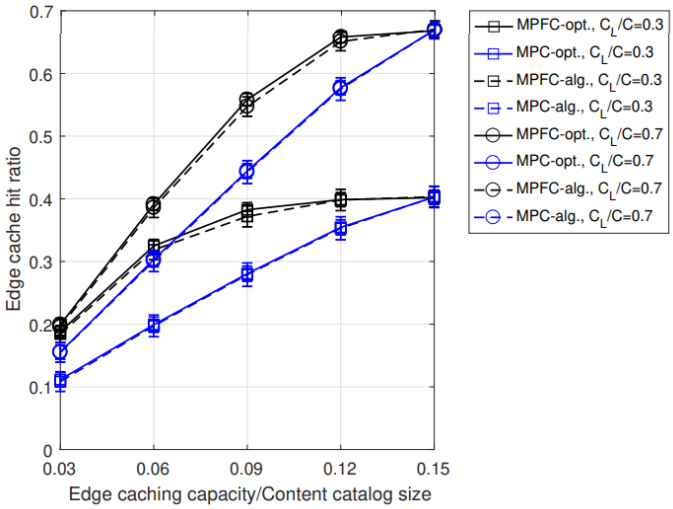

Fig. 4. Average content retrieval latency [12]

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 aproach not concidering the lifetime of the data into the caching decisions. In all three scenarios Most Popular Fresh Contents (MPFC) achieved lower content retreival latency, higher cache hit rates and higher freshness probabilities. In figures IV-A, IV-A and IV-A 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 freshnes probability, with increasing cache size, the freshness probability of MPFC equals to MPC.

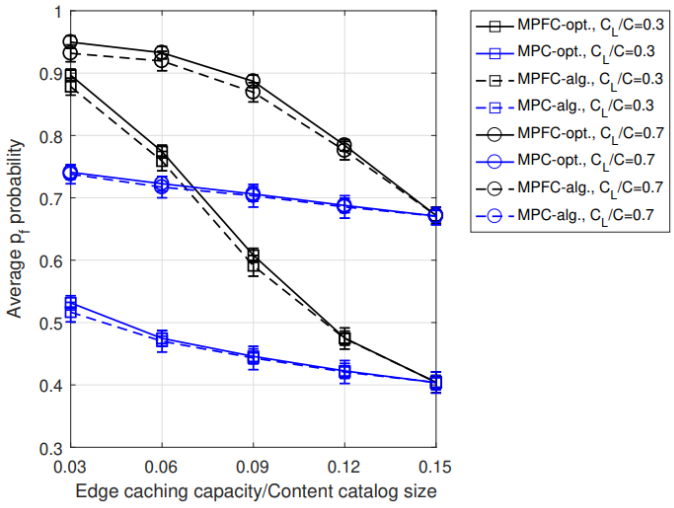

Fig. 5. Average content retrieval latency [12]

B. *Moth-flame optimization algorithm*

In [4] from 2023 the authors propose a caching strategy using the Moth-Flame Optimization (MFO). It is a population-based metaheurist 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 are using the algorithm to group similar content and determine the optimal locations for caching certain data. They also cluster the edge nodes and select cluster heads using their propsed algorithm Moth-Flame Optimization-Cluster Head Selection (MFO-CHS). It furhter reduces the latency and energy consumption in the network. In their architecture, also uses SDN with a centralized SDN Controller. For southbound communication it also uses the OpenFlow protocol and for the northbound interface RESTful APIs. 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 intervall 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 the content freshness and the content popularity as features to make caching descions. From these two features, a caching weights are calculated, which are then used by the MFO-EC algorithm. To create a efficient environment to be able to process massive amounts of data, the authors propse to use IoT clusters. That should bring a better quality of experience to users. Edge networks are created to be able to better use and share their computational power and resources. With the SDN Controller this complexity is hidden to the users. The MFO-CHS algorithm is used to dynamically select the cluster heads. 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

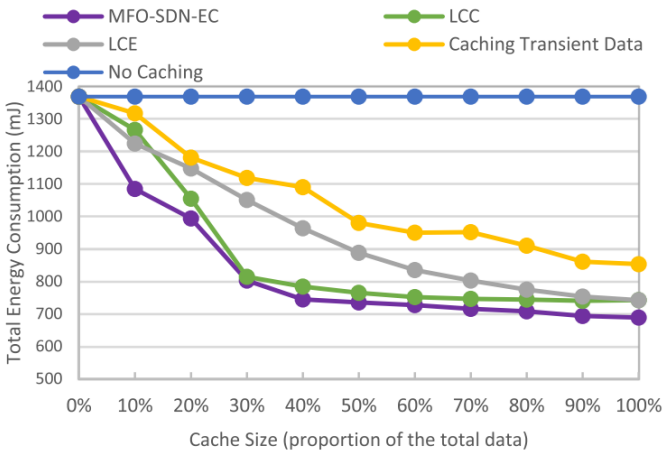

Fig. 6. Total Energy Consumption (mJ) VS cache size [4]

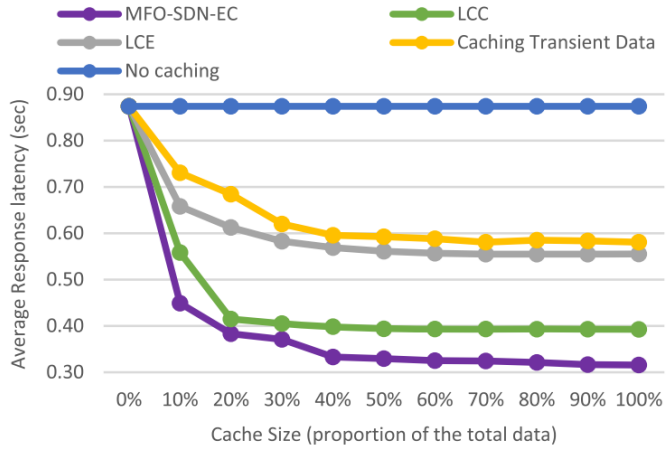

Fig. 7. Average Response latency (sec) VS cache size [4]

network's energy consumption is at the minimum.

V. Conclusion

SDN is great. It brings a lot of advantages...

References

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

[2] 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.

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

[4] 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.

[5] 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.

[6] 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).

[7] 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.

[8] 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.

[9] 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.

[10] Alexander Nunez et al. *A Brief Overview of Software-Defined Networking*. 2023. arXiv: 2302 . 00165 [cs.NI]. URL: https://arxiv.org/abs/2302.00165.

[11] 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.

[12] 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.

[13] Ali Haider Shamsan and Arman Rasool Faridi. "SDN-Assisted IoT Architecture: A Review". In: *2018 4th International Conference on Computing Communication and Automation (ICCCA)*. 2018, pp. 1–7. DOI: 10.1109/CCAA.2018.8777339.

[14] 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.