


Article

Cache Servers Placement Based on Important Switches for SDN-Based ICN

Jan Badshah ^{1,†}, Majed Mohaia Alhaisoni ², Nadir Shah ^{1,*,†}  and Muhammad Kamran ³

¹ Department of Computer Science, COMSATS University Islamabad, Wah Campus, Islamabad 45550, Pakistan; janbadshahsafi@gmail.com

² Department of Computer Science, College of Computer Science and Engineering, University of Hail, Hail 2440, Saudi Arabia; majed.alhaisoni@gmail.com

³ Department of Cyber Security, College of Computer Science and Engineering, University of Jeddah, Asfan Road, Jeddah 21577, Saudi Arabia; m.kamran.nuces@gmail.com

* Correspondence: nadirshah82@gmail.com

† These authors contributed equally to this work.

Received: 21 November 2019; Accepted: 24 December 2019; Published: 27 December 2019



Abstract: In centralized cache management for SDN-based ICN, it is an optimization problem to compute the location of cache servers and takes a longer time. We solve this problem by proposing to use singular-value-decomposition (SVD) and QR-factorization with column pivoting methods of linear algebra as follows. The traffic matrix of the network is lower-rank. Therefore, we compute the most important switches in the network by using SVD and QR-factorization with column pivoting methods. By using real network traces, the results show that our proposed approach reduces the computation time significantly, and also decreases the traffic overhead and energy consumption as compared to the existing approach.

Keywords: SDN-based ICN; centralized cache management; SVD; QR factorization

1. Introduction

Recently, Internet users are producing most of the content-centric traffic, and the amount of this traffic is on the rise due to the popularity of video streaming services among the users, such as YouTube and Netflix [1]. The Cisco's Visual Networking Index reported that, due to an exponential increase in the demand for video streaming, mobile data communication traffic would become sevenfold by 2021 [2]. In this traffic, video traffic will be 78% of the total mobile traffic. The increasing demand for video streaming of public users through mobile devices, the Internet speed is significant to fulfill the desired need for end-users by efficient use of bandwidth. The original TCP/IP (Transmission Control Protocol/Internet Protocol) model was not enough to handle these services efficiently because the TCP/IP model uses host-based routing, which does routing and forwarding of data packets based on hosts (i.e., based on host IP (Internet Protocol) address). Addressing this problem, researchers proposed information-centric networking (ICN) by implementing the content-based routing at the network layer instead of host-based routing [3].

ICN is also called named data networking (NDN). A detail working of ICN is that a source node requests an object by initiating a request message. Named-Data-Object (NDO) specifies the object. The message is forwarded based on NDO to the destination nodes having a copy of the object specified by NDO. Upon receiving the request for the object, the destination node forwards the object in the response message to the source according to the ICN routing algorithm. In the ICN routing algorithm, each node publishes its shared objects in the network. Each forwarding node also stores a copy of the object present in response message in the cache. The reason for this is that the forwarding node can

satisfy subsequent requests for the same object without forwarding to the original destination node where the object is stored.

The traditional computer network, the one used in practice, is distributed in nature [4,5]. More specifically, the traditional computer network implements both the control and data planes in each forwarding device (i.e., routers and switches). The distributed nature of the traditional computer network causes several limitations, e.g., the traditional computer network is challenging to manage and control [6]. More specifically, the vendors implement (mostly embedded in the hardware) both the data and control planes in forwarding devices of the traditional computer network. In this closed system, one can not experiment with the new ideas and innovations due to these hard-coded logics in these devices [6]. Moreover, traditional computer networks are surprisingly fragile and difficult to manage. Because these are managed using a low-level configuration of devices, for example, policies are implemented mostly on these devices; like blocking a user, we need ACL (Access Control List) entry to be added in the list and knowing the user IP address. To overcome these issues of the traditional network, recently, Software Defined Network (SDN) has emerged as a new networking architecture separating the control plane from the data plane of forwarding devices. Due to this decoupling of the control plane from the data plane, SDN has several advantages over the traditional network, like ease of traffic engineering and enforcing security policies. The control plane in SDN is implemented at the logically centralized controller, and the data plane is implemented at forwarding devices. The centralized controller communicates with forwarding devices using northbound API (Application Programming Interface), e.g., OpenFlow Protocol. Thus, SDN is a promising technology to make network programmable, easy to control, and manage.

Due to this paradigm shift in computer network architecture, the ICN approaches proposed for the traditional network, like the one in [7], cannot be implemented in SDN. The reason is that the traditional network is distributed in nature, while SDN is centralized architecture. The researchers have suggested several approaches to support ICN functionalities in SDN; we call these approaches as SDN-based ICN, addressing different issues like cache management, routing, communication between the controller and ICN-enabled router, and name-based forwarding [3–5,8–26]. For cache management in SDN-based ICN, two types of approaches are devised as follows:

- (a) Pervasive caching approach: In this approach, all forwarding devices have the capabilities of content caching according to caching policies [7]. The main advantage of such an approach is its higher resilience to node failure and short end-to-end delay for data delivery. However, the disadvantage of this approach is that it caches the content redundantly at many forwarding devices. This phenomenon, in turn, results in inefficient management of cache storage. An example approach of the pervasive caching is [4].
- (b) Centralized caching approach: In this approach, the content caching storage is not available in every forwarding device, but, instead, there are one or a few dedicated cache server(s). This approach has several advantages as follows:
 - i First, this approach manages the cache storage efficiently by avoiding caching a content redundantly.
 - ii Second, this approach makes communication faster as follows. In a pervasive caching approach, a forwarding device does two primary jobs: routing and content caching. This phenomenon makes a forwarding device more complex and reduces its performance in pervasive caching. The centralized caching approach decouples content caching from the forwarding devices, and the cache server performs content caching. This phenomenon increases the performance of forwarding devices.
 - iii Third, the authors in [8] show that the centralized caching approach has a higher cache hit ratio as compared to the pervasive caching approach. The reason is that the cache storage avoids storing a large number of redundant contents. It makes space for a greater number of unique contents in the cache storage.

- iv Fourth, this approach reduces the installation cost of the network [8].

Moreover, the centralized caching approach is also advocated in many other networking applications as follows:

- i By installing the flow rules proactively for all the possible flows as the network gets running, this can lead to the overflow of the flow table at the switches. To solve this problem, DIFANE [27] advocates a centralized caching approach for storing the flow rules generated proactively by the controller. More specifically, DIFANE attempts to offload the controller by generating the flow rules for all possible flows proactively as the network gets running. Then, the controller distributes these flow rules disjointly among the authoritative servers (we can call them cache servers). When a flow arrives at the switch, and the switch does not have the flow rules for the flow, then the switch asks one of the authoritative servers instead of the controller for the flow rules. This phenomenon offloads the controller from these flows. Furthermore, the authoritative servers are attached to the switches; thus, DIFANE uses an in-band communication model [28].
- ii For a Server and Network Assisted DASH (SAND) architecture used for the video streaming applications, the authors in [29] combine both SDN and NFV (Network Functions Virtualization) technologies by creating virtualized caches using NFV hosted by the servers connected to the switches. The authors deploy the virtualized caches in the network by considering the number of online requesting hosts (clients), bandwidth of the paths, and the locations in the network.
- iii The OFELIA project [14] places the cache server outside the SDN switch in order to support the ICN functionality in SDN.

The recent approaches proposed for centralized cache management in SDN-based ICN do not place the cache servers efficiently as follows. The authors in [8] deploy the cache servers based on minimum path-stretch values. However, it produces more traffic overhead. To reduce the traffic overhead, Badshah et al. [28] suggest deploying the cache servers by considering higher closeness centrality, minimum path-stretch values, and higher betweenness centrality of the network. However, when the number cache servers get increased, the time taken to compute the location of cache servers using the values of closeness centrality, path-stretch, and betweenness centrality is very computationally extensive due to the NP (non-polynomial) nature of the problem. We observe that the computation time of this approach, i.e., the existing one in [28], for two cache servers and three cache serves are about 9 h and 1 month, respectively, in the topology of 415 switches using the real network traces (<https://github.com/fg-inet/panoptisim> Accessed on 4 July 2019) on the desktop computer of Intel CPU i3 which runs Fedora Linux 23. We solve this problem by using the concept of important switches as follows.

Through experimental results, the researchers show that the traffic matrices in the networks are approximately low-rank [30], i.e., the traffic matrices are sparse. This phenomenon means that most of the data flows are passing through only a few switches/routes. We call these switches as important switches in the network. The sparse matrices can be solved by using singular-value-decomposition (SVD) and QR factorization with column pivoting techniques of linear algebra [31]. We describe in detail these steps in Section 3.2. Our proposed approach reduces the computation time to a few seconds for our topology of 415 switches. After computing the important switches, this paper proposes to place the cache servers on the most important switches according to the number of required cache servers. Through simulation on the real network traces, the results show that this approach reduces not only the computation time, but also performs better than the existing approach [28] in terms of traffic overhead in the network.

We organize the rest of the paper as follows. We discuss the related work in Section 2. Section 3 explains the proposed approach in detail. The results are discussed in Section 4. Finally, Section 5 concludes the paper along with future research directions.

2. Related Work

We are going to discuss the working of some of the related existing approaches in this section. Michael et al. [10] extended Home Router Sharing based on Trust (HORST) [32] by adopting HORST in SDN architecture. HORST [32] is an existing system to share among trusted users the access to Wi-Fi. Moreover, by utilizing the information of users from their online-social-network (OSN) profiles, HORST pre-fetches videos and caches at home routers. HORST also interconnects home routers in the peer-to-peer (P2P) overlay. This paper uses an SDN controller for content distribution and request forwarding in HORST. Furthermore, the authors introduced three tiers architecture for content sharing by using home router, caches within Internet Service Provider (ISP), and at the content distribution network (CDN) as follows. At tier-1, their proposed approach stores the data in the data center of content provider. At tier-2, their approach caches the data at edge networks and ISPs using the CDN fashion. At tier-3, their proposed approach utilizes the home routers of the user network to cache data by using HORST. Their proposed approach forwards the request of the user for video content to the closest cache in such a way to improve the quality of service and the quality of experience, and reduces the inter-domain traffic for ISP networks. However, their proposed approach does not consider path stretch, closeness centrality, and other parameters for cache server placement.

Kalghoum et al. in [11] suggested a novel architecture Forwarding-Content-Caching and Routing Name Data Networking based Software Defined Networking (FCR-NS). This model resolves the three main issues related to forwarding strategy, speed of router, and cache replacement policy. The authors state that the existing architectures are using the best-route forwarding strategy that causes wastage of different resources. The reason behind the slow processing is the visiting of the Content Store (CS) table, Pending Interest Table (PIT), and Forwarding Information Base (FIB) table for searching the request message and lack of efficient cache replacement policy. The authors resolved the problems as mentioned above by using both the concept of decoupling of SDN and set bloom filter tables. The efficient cache replacement strategy is calculating the switches' popularity on a real-time basis. The SDN controller manages the fast-intra-zone routing, and the bloom filter (BF) tables are using to manage the fast inter-zone routing process. The default policy for content caching is leaving a copy of the request message on all routers on the path. This policy leads to content redundancy in the network and resource consumption. The authors suggest an architecture of several zones, as shown in Figure 1, and these are further divided into two main planes, the data plane and centralized SDN controller, which is further composed of four tables.

The proposed FCR-NS has four tables in the Controller and Five Tables in the FCR-NS switch. The controller tables are Global-Top table, Global-Data table, Routing Information Base (RIB) table, and Global-Data-BF table. However, the FCR-NS switch is composed of five tables that are CS table for caching, CS-BF table, FIB table, FIB-BF table, and PIT table. The authors proposed a replacement policy for content caching that considers the popularity of data. The replacement takes place based on the FIB table, which stores the record of popular content. Furthermore, the authors improve the routing by dividing of routing into intra-zone and inter-zone. The FIB tables of switches in intra-zone and contains popular data the network, which is used for the fast reply of request messages from the nearest nodes. The controller uses Global-Data and Global-Top tables for the construction of RIB and a table for the storage for all data in that network. Then, the controller constructs the FIB table for each switch in the network. The Dijkstra routing algorithm is used to perform this activity and runs for the addition of new data in the Global-Data table and updates the switches of the zone. The inter-zone routing in this approach works as follows. If the node response message is pending in a zone, the node forwards a request message to the controller and delivers to the requester in case of availability in the controller zone. Otherwise, the request message is forwarded to other controllers of the network. After receiving the request message from another controller, the controller checks the Global-Data-BF table and sends the response message back to requesting controller. This process continues among the controller of all zones. The Global-Data-BF bloom filter plays a basic role to speed up and make more efficient the search between zones.

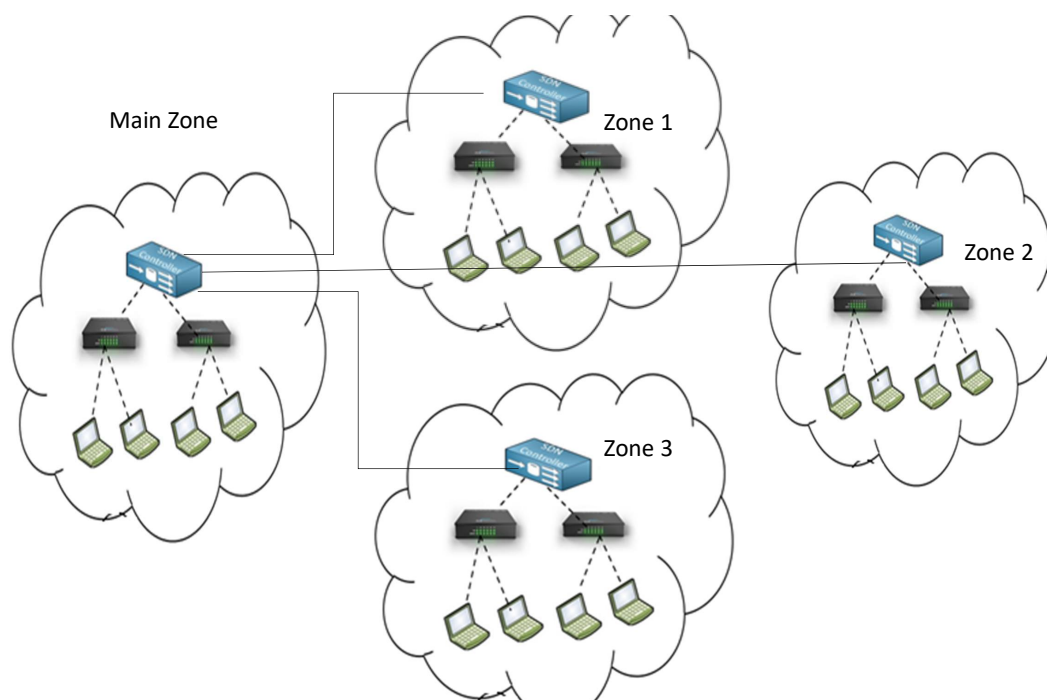


Figure 1. The FCR-NS architecture [11].

The authors in [14] implemented a new architecture, called OFELIA, as shown in Figure 2. The OFELIA project is a pan-European experimental platform open to researchers based on OpenFlow. In the proposed architecture, “Inter Working Elements” (IWE) in the ICN node translates regular ICN format of non-Openflow to Openflow format and vice versa. Cache server is implemented outside the SDN switch (OF-switch). This approach can also be called the centralized caching approach.

Kim et al. [8] suggested a centralized caching approach instead of maintaining a cache at each router in the network for ICN. Their proposed approach distributes the communication overhead between the router and cache server by the reduction of inter-domain traffic due to a higher hit ratio at the cache server as compared to the pervasive caching approach. Second, in the pervasive caching approach, if a router is processing one request, then other packets will wait. If other packets belong to real-time traffic, then they will get delayed and may become useless. However, in their proposed centralized cache server approach, the probabilities of such occurrences are reduced to offloading the cache lookup from the routers. Figure 3 depicts the working of their proposed approach, and its detail is as follows.

The proposed approach deploys the cache server such that at least one router between two edge routers of a domain should be connected to the cache server. It is a set cover problem that is NP-Complete. To reduce this computation complexity, the authors choose the routers which exist in most paths (in the shortest paths among the edge routers). Therefore, the proposed approach is placing the cache server based on path stretch value. This produces more routing overhead in the network. To address this problem, the authors in [28] place the cache servers in the network based on the optimum value of closeness centrality, path-stretch, and betweenness centrality values. However, this is an optimization problem for placing the cache server based on the optimum value of path-stretch, betweenness centrality, and closeness centrality. More specifically, when the number of cache servers gets increased, the time taken to compute the location of cache servers using the values of closeness centrality, path-stretch, and betweenness centrality becomes very computationally extensive due to NP (non-polynomial) nature of the problem. To address this problem, we suggest placing the cache servers in the SDN-based ICN based on the important switch/router concept; we describe its detail in the following section. A comparison of the existing approaches described above are given in Table 1.

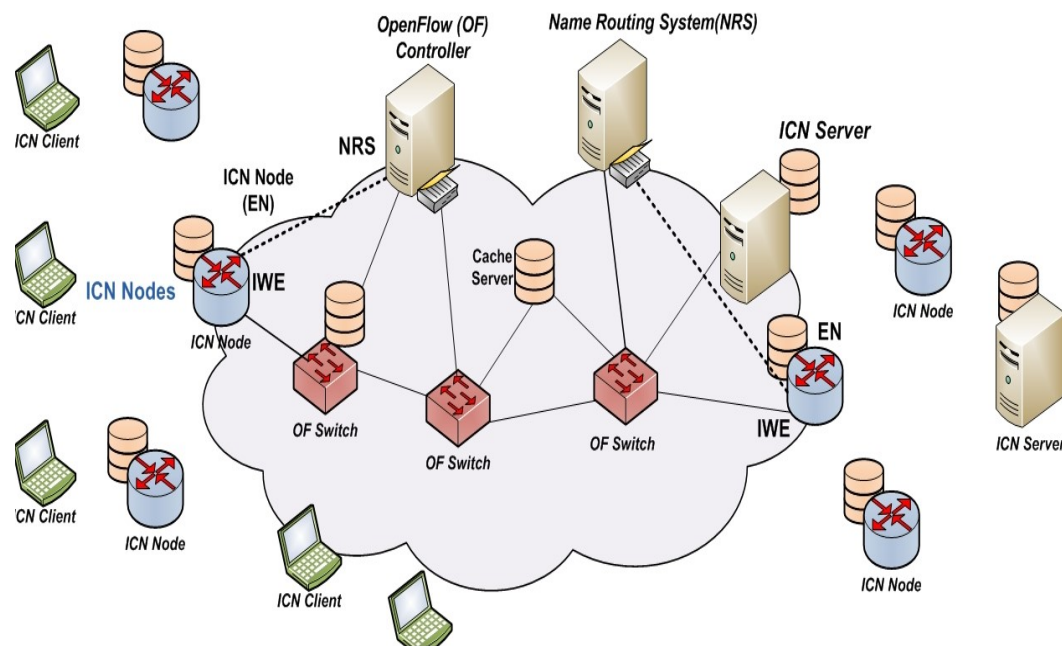


Figure 2. Details of OpenFlow solution to support ICN [14].

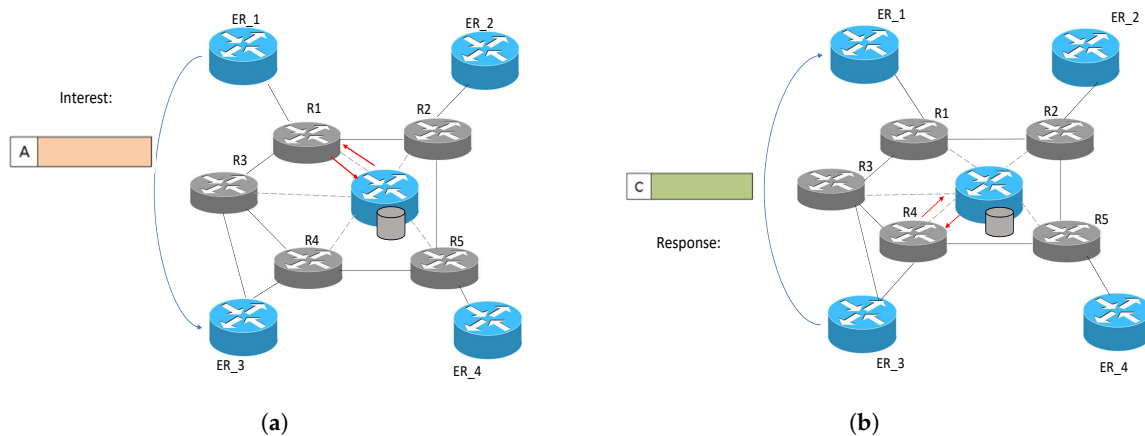


Figure 3. The working of the proposed approach in [8]; (a) interest forwarding; (b) response forwarding.

Table 1. The comparison of existing approaches for SDN-based ICN.

Approach	Centralized/Pervasive Caching	Cache Replacement Policy	Single/Multiple Controller(s)	Cache Deployment
[10]	Pervasive	Not Discussed	Single	Random
[11]	Pervasive	Popularity-based	Multiple	Random
[14]	Centralized	Not Discussed	Multiple	Random
[8]	Centralized	Not Discussed	Multiple	Path-Stretch
[28]	Centralized	IFIFO	Multiple	Path-Stretch & closeness & betweenness

3. Proposed Work

In our proposed approach, we call it Cache Server Placement based on Important Switches (CSP-IS), which contain the following components.

3.1. Traffic Matrix Computation

We assume that the network is running for some time (may be days or weeks) and has the data traffic log. From this traffic, we store the number of flows passing through the switches at different time intervals in a matrix $M \in \mathbb{R}^{t \times s}$, where t is the number of rows representing the number of consecutive time intervals, and s is the number of columns representing total number of switches in the network. More specifically, a row i represents the total number of flows passing through all switches at the interval i , and a column j denotes the time series of number of flows passing through j th switch in the network. It is further assumed that $t \gg s$. After obtaining the traffic matrix, we compute the important switches, as described in the following section.

3.2. Computing Important Switches

As mentioned before that the traffic matrices of the network are spars, we can solve it by using the SVD and QR factorization with column pivoting techniques of linear algebra as follows. First, our proposed approach applies SVD function (technique) on the matrix M in order to show how SVD can reveal the spatial correlation. After applying the SVD on the matrix M , it decomposes the matrix into three matrices as given in Equation (1):

$$M_{t \times s} = U_{t \times t} S_{t \times s} V_{s \times s}^T, \quad (1)$$

where $U \in \mathbb{R}^{t \times t}$ and $U^T U = I$, $V \in \mathbb{R}^{s \times s}$ and $V^T V = I$, and $S_{t \times s}$ is a diagonal matrix with diagonal values; they are, say, represented by $\{\sigma_i | i \in \{1, \dots, t\}\}$ in descending order (i.e., a diagonal value $\sigma_i \geq \sigma_{i+1}$). These diagonal values are also called singular values and represent the importance of a switch in the matrix M . $U_{t \times t}$ is known as the left singular matrix. $U_{t \times t}$'s columns, say they are represented as $\{u_i | i \in \{1, \dots, t\}\}$, represent the left singular vectors and are orthogonal to each other. $V_{s \times s}$ is the right singular matrix. $V_{s \times s}^T$'s columns, say they are represented by $\{v_i | i \in \{1, \dots, s\}\}$, are known as the right singular vectors and are orthogonal to each other.

The relation given in Equation (2) is the crucial property of SVD:

$$M_{t \times s} = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_s u_s v_s^T. \quad (2)$$

It is worth noting that a singular value can be $\sigma_i \geq 0$ in Equation (2). Suppose that we have q number of positive singular values out of s number of singular values. Then, one can infer from Equation (2) that every column of M can be computed by a linear combination of q left singular vectors, $\{u_i | \sigma_i > 0, i \in \{1, \dots, q\}\}$, and every one of these vectors u_i has its coefficient of a positive singular value. The q is called the rank of M . These q left singular vectors are orthogonal basic vectors and spanning the column space of M . This phenomenon indicates that each M 's column represents a linear combination of these q basic vectors. In other words, the q basic vectors can be recovered using exactly the q columns of M [31]. We call such q columns in M as its basic columns. Thus, by considering the traffic matrix $M \in \mathbb{R}^{t \times s}$ of rank q , we compute its q basic columns, which represent the q basic switches in our target scenario. Therefore, the traffic information passing through each switch is completely represented using the linear combination of the traffic information passing through such q basic switches. In other words, we can completely and correctly represent the matrix M through these q basic columns. For collecting the traffic information passing through the switch, the rank q of a traffic matrix shows that traffic information passing through q switches can be used to completely and correctly recover the global traffic matrix. Moreover, the traffic matrix M of a network usually also has a lower rank, say k , which is also known as effective rank and $k \ll q$. This indicates that the set of singular values are sparsely distributed. In other words, for traffic matrix M , we have only a few large singular values, and other singular values have minimal values.

To explain this concept, let consider Equation (2) again. This equation shows that the information of the matrix M is distributed into s terms in the right-hand side of the equation. Each term $\sigma_i u_i v_i^T$ is

the product of a coefficient σ_i and a matrix $u_i v_i^T$. Suppose that the first k terms are very large singular values, where $k \ll q$, and the rest are small singular values. This means that the first k terms in Equation (2) concentrates a major portion of M 's information. This, in turn, indicates that only k basic vectors can represent the whole matrix M . In comparison with the considering all columns of M and using only q basic columns, the approximated representation of M using only k basic columns decreases the computation time by decreasing a large number of required basic columns. Now, we have to find $k \ll q$ basic columns of M such that they can be used to approximately represent M . This problem belongs to subset selection problems in linear algebra [31] and is NP-complete. There is no algorithm to determine its optimal solution. However, our proposed approach use an approximation solution in [31] to compute a subset of k columns of M that represent the space formed by its first k basic vectors $\{u_i | i \in \{1, \dots, k\}\}$. The first k basic vectors represent the basic vectors u_i in the first k terms in the right-hand side of Equation (2). The switches associated with the k basic columns in M are called basic or important switches. We explain the subset selection problem to find k basic columns as follows.

Problem 1. For the matrix $M_{t \times s}$ and the integer k , the subset selection problem is to find the permutation matrix P such that

$$MP = [M1_{t \times k} \ M2_{t \times (s-k)}], \quad (3)$$

where $M1_{t \times k}$ and $M2_{t \times (s-k)}$ are matrices. The goal is to contain maximum information and to let all the columns sufficiently independent in $M1_{t \times k}$.

To determine all the columns sufficiently independent of $M1$, it means to maximize the smallest singular value of $M1$. For this purpose, first, our proposed approach performs SVD on M as given in Equation (1). After this, our proposed approach multiplies the matrix P with the right singular matrix V and gets:

$$P^T V = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix},$$

where $V_{11} \in \mathbb{R}^{k \times k}$. In case V_{11} is non-singular, we can obtain the lower bound on the smallest singular value $\sigma_k(M1)$ of $M1$ as follows:

$$\sigma_k(M1) \geq \frac{\sigma_k(M)}{\|V_{11}^{-1}\|}, \quad (4)$$

where $\sigma_k(M1)$ represents the k th (smallest) singular value of $M1$. By maximizing $\sigma_k(M1)$, it is equivalent to maximizing the lower bound of $\sigma_k(M1)$. This is equivalent to minimizing $\|V_{11}^{-1}\|$ in Equation (4). To solve this problem, then our proposed approach applies the QR factorization with column pivoting [31] on M . Algorithm 1 describes the overall solution. After computing the important switches, we configure the cache servers to the important switches and do the necessary configuration; we describe the procedure in Section 3.3.

Algorithm 1 Algorithm for Finding Important Switches.**Input:** matrix $M_{t \times s}$ and integer k **Output:** $MP = (M1, M2)$ where $M1$ is matrix of $t \times k$ **Step1:** Apply SVD over $M_{t \times s}$ using Equation (1)**Step2:** Apply QR factorization with column pivoting to M using following Equation:

$$M = QRP^T$$

Here, Q , R , and P are matrices. P is a permutation matrix.**Step2:** Perform permutation to all columns of M through multiplication of M with P . After this, the first k columns of M is $M1$ as given below

$$MP = [M1_{t \times k} \ M2_{t \times (s-k)}]$$

3.3. Configuring Cache Server Information

After computing the k number of the most important switches as described in Section 3.2, we place these cache servers by attaching a cache server with one of the important switches at a time. After placing the cache servers in the network, our proposed approach configures the location information at the data plane by using the access control list (ACL) commands at the controller. Then, the controller forwards the flow rules to all forwarding devices by instructing the forwarding devices to forward a request for content to one of the closest cache servers. As the network gets configured, we assume that every content provider publishes the information of its all contents at the controller. Thus, the controller has the global view of which content is available at which content provider. When a host wants to retrieve the content (say C1), the host forwards the content request to the first switch. After receiving a content request, a switch forwards the content request to the nearest cache server (as the controller at each switch already configures this). When the cache server (say CS1) receives the content request, CS1 looks for the matching in its cache. If CS1 finds the matching, then the content reply is sent to the requesting host. Otherwise, CS1 forwards the content request to the controller. After receiving the content request, the controller computes the best path to the content provider and installs the path in the data plane. After this, the controller forwards the content request to the content provider. When the content provider receives the content request, the content provider looks for in its list of shared content for matching. If the content provider finds the matching, then the content provider sends the content reply on the reverse path established by the controller toward the CS1. After receiving the content reply, CS1 stores a copy of the content with itself and forwards the content reply toward the requesting host. If the cache is full, then first-in-first-out (FIFO) replacement is used by the CS1. We divide content into chunks, and a node transmits a chunk at a time.

4. Results and Analysis

For performance comparison, we took the network traces from the website (<https://github.com/fg-inet/panoptisim>). We assume that there are 2590 hosts that are accessing/sharing 10,000 files. The traces have 415 forwarding devices. Furthermore, we assume a file of size 10^6 chunks and a chunk of 1500 bytes. Moreover, we assume that the cache server can process 1200 content requests per second and storage capacity of 4×10^8 chunks. Our simulation runs on desktop Intel CPU i3, which runs Fedora Linux 23. The existing scheme [28] deploys the cache servers using the values of path-stretch, betweenness, and closeness centralities. We analyze the performance comparison of the existing approach [28] and our proposed approach in terms of computation time and traffic overhead. We present the results considering the number of cache servers from 1 to 3, and the number of sources requesting the contents from 1000 to 1800. The traffic overhead shows the number of transmissions at the network layer. Table 2 shows that the computation time of the existing approach

increases as the number of cache servers is increasing. In our assumed simulation environment, it took about a month to compute the location for three cache servers. However, the proposed approach can compute the importance of all switches in a few seconds. Our proposed approach reduces not only the computation time, but also reduces the traffic overhead when there is more than one cache server as indicated in Figures 4–6. From these figures, one can also note that the reduction of traffic overhead increases as the number of sources increases. Thus, our proposed approach solves the problem by reducing not only the computation time, but also decreasing the traffic overhead in the network.

Table 2. The comparison of computation time in hours.

Number of Cache Servers	Proposed Approach	Existing Approach
1	0.001111111 h	1.887333333 h
2	0.001111111 h	20.3 h
3	0.001111111 h	1492.323001 h

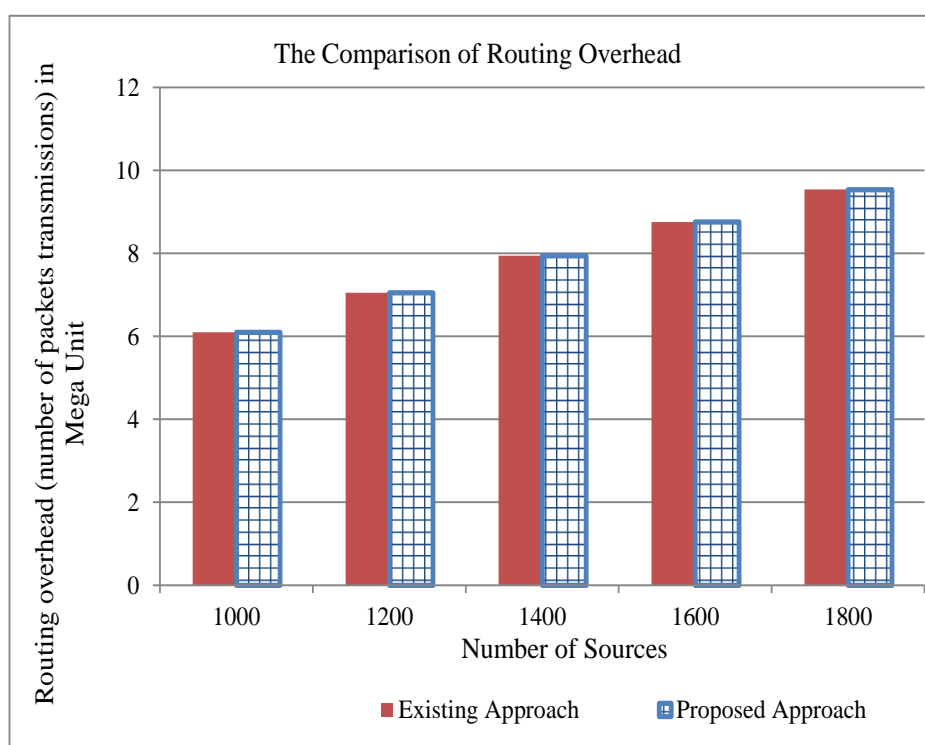


Figure 4. The comparison of traffic overhead of the existing and proposed approaches for one cache server.

Energy consumption is an important parameter for evaluating a protocol of computer networking. As we have stated above that our proposed approach reduces traffic overhead, this can also reduce the energy consumption in the network and, consequently, can achieve the objective of green computing. We use the energy model described in [33] to show the results of energy consumption for per-byte transmit and receive, and per-packet processing by varying both the number of cache servers and the number of flows. We show the reduction of energy consumption as compared to the existing approaches in order to show the significance of our proposed approach. The traffic overhead of both our approach and the existing approach is the same, as shown in Figure 4. Therefore, the difference between the energy consumption of our proposed approach and the existing approach is zero, and we do not show it in the graph. Figures 7–9 show that our proposed approach reduces significantly the per-byte transmit and receive energies, per-packet processing energy, respectively, for two cache servers. Similarly, one can note from Figures 10–12 that our proposed approach reduces more per-byte transmit and receive energy, and per-packet processing energy when we increase the number of cache

servers from 2 to 3. The reason is that, when we have more number of cache servers in the network, then the path length from a content requester to the closest cache servers is decreased. This reduces the number of transmissions and the energy consumption.

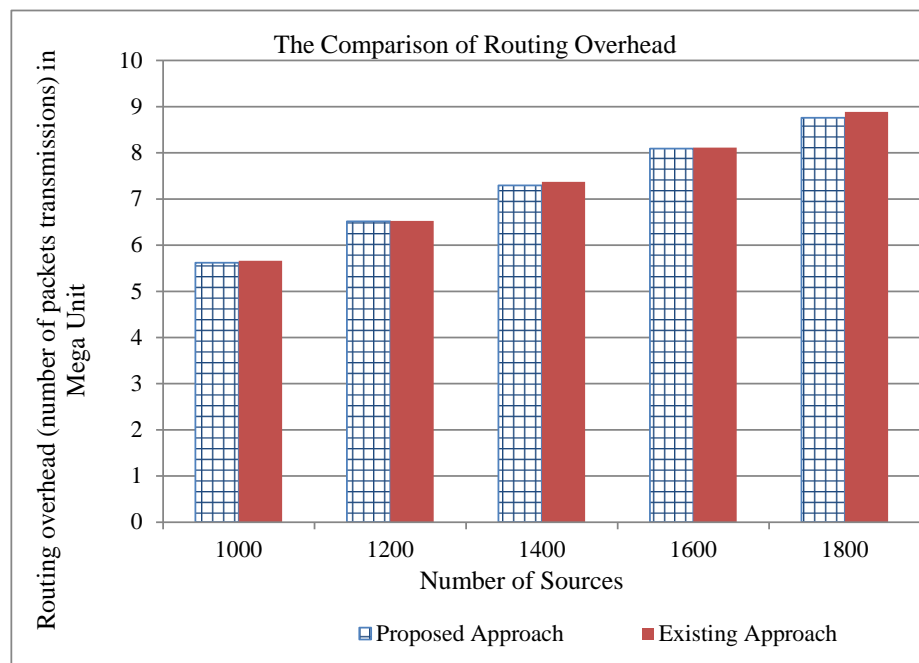


Figure 5. The comparison of traffic overhead of the existing and proposed approaches for two cache servers.

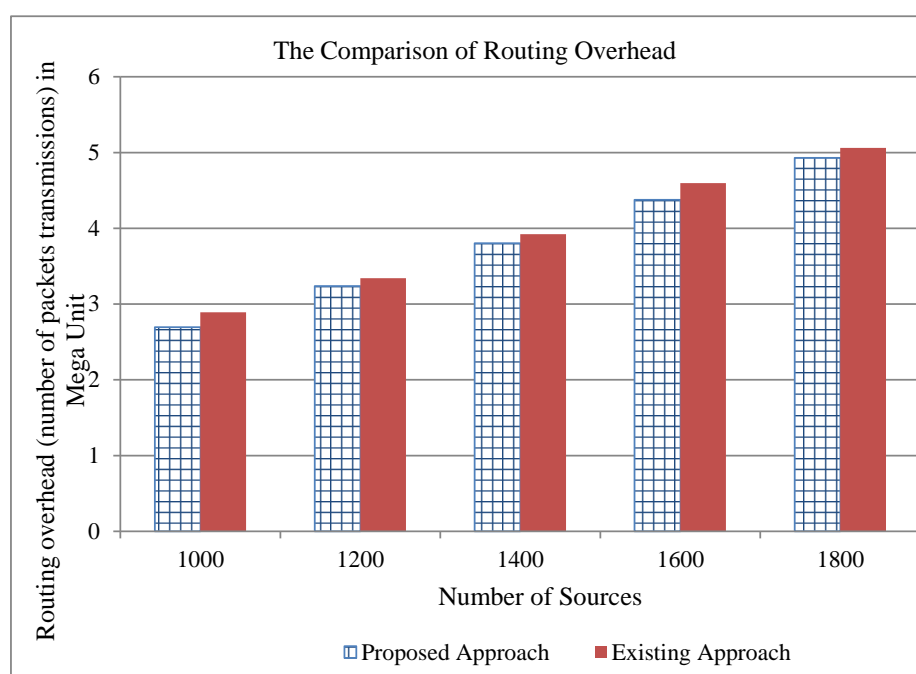


Figure 6. The comparison of traffic overhead of the existing and proposed approaches for three cache servers.

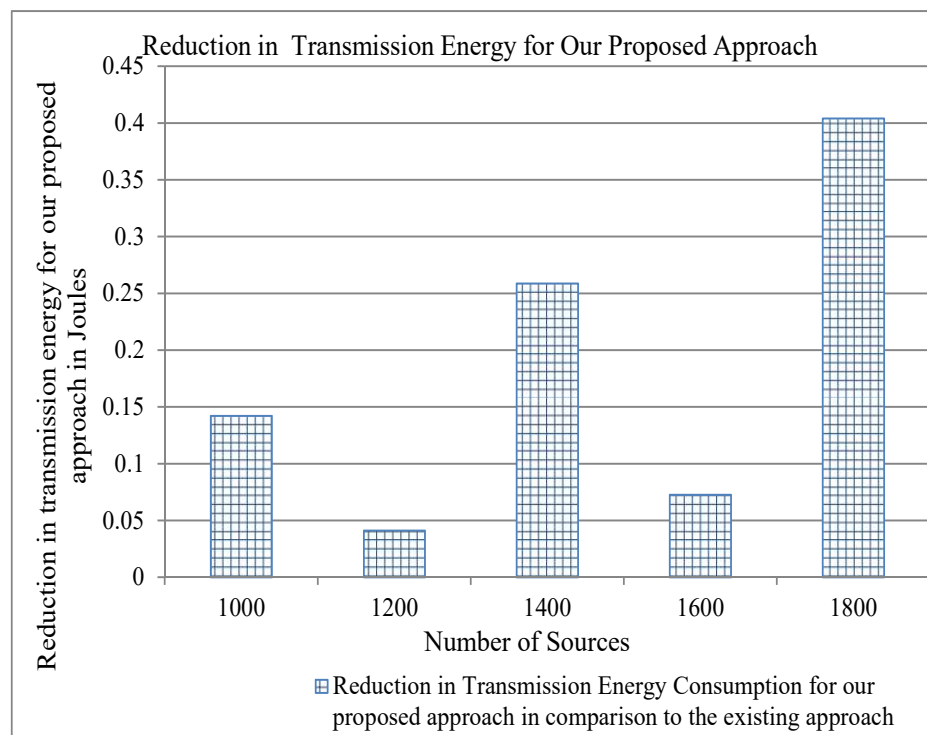


Figure 7. Reduction in Transmission Energy for our proposed approach in comparison to the existing for two cache servers.

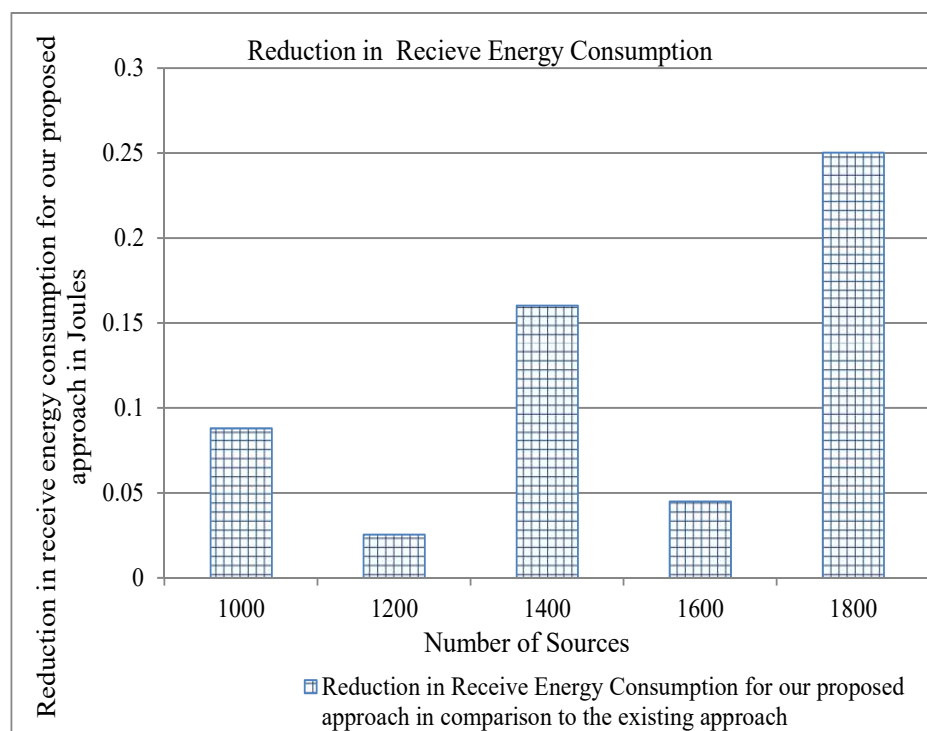


Figure 8. Reduction in Receive Energy for our proposed approach in comparison to the existing for two cache servers.

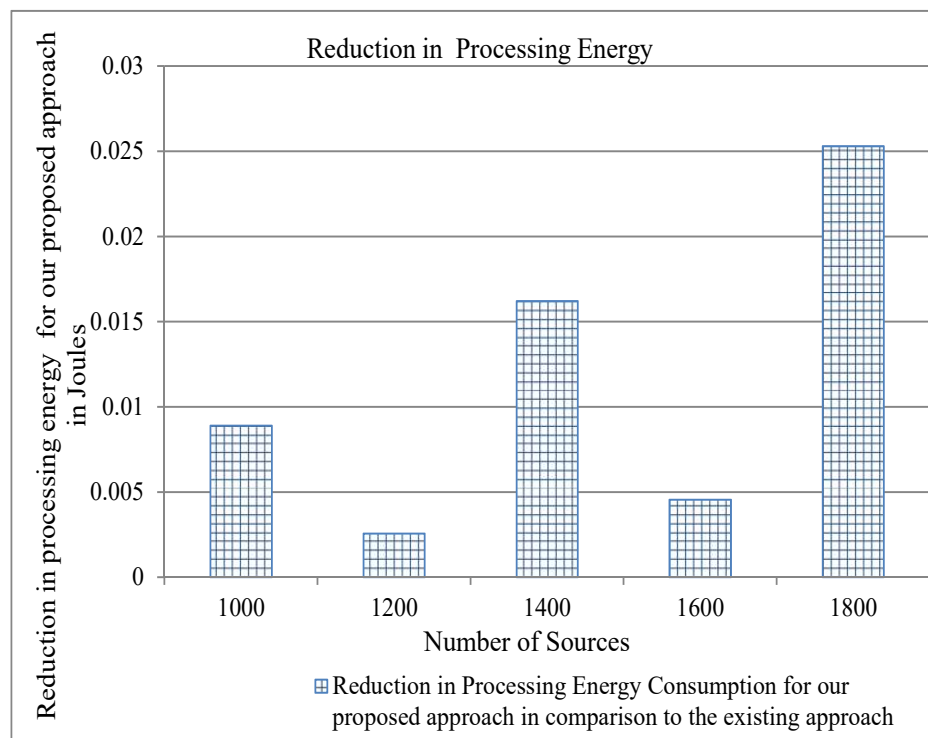


Figure 9. Reduction in Processing Energy for our proposed approach in comparison to the existing for two cache servers.

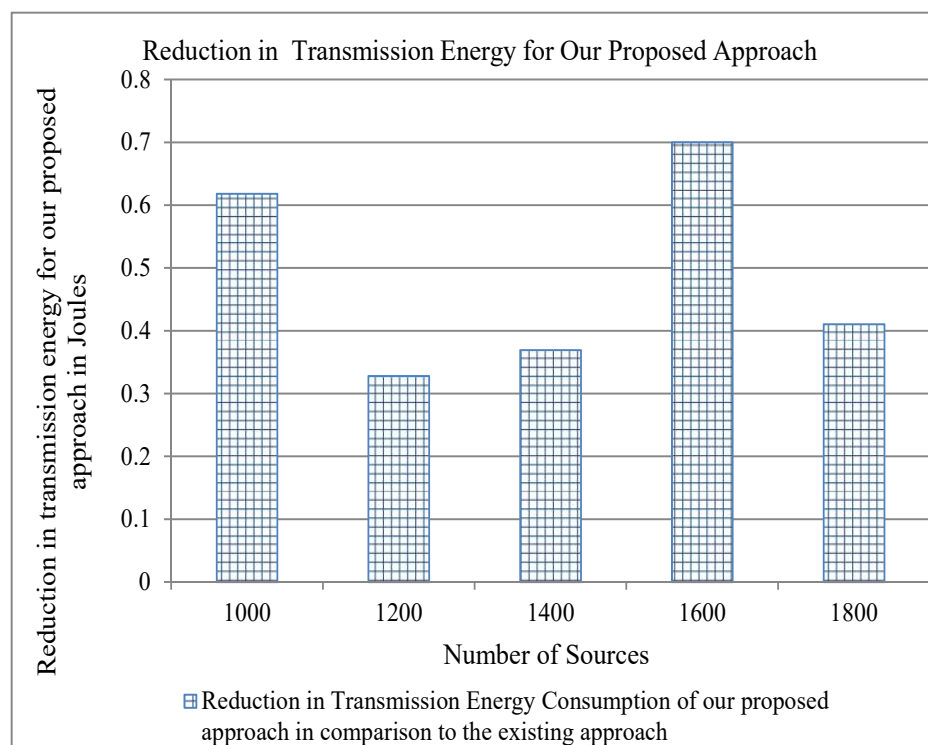


Figure 10. Reduction in Transmission Energy for our proposed approach in comparison to the existing for three cache servers.

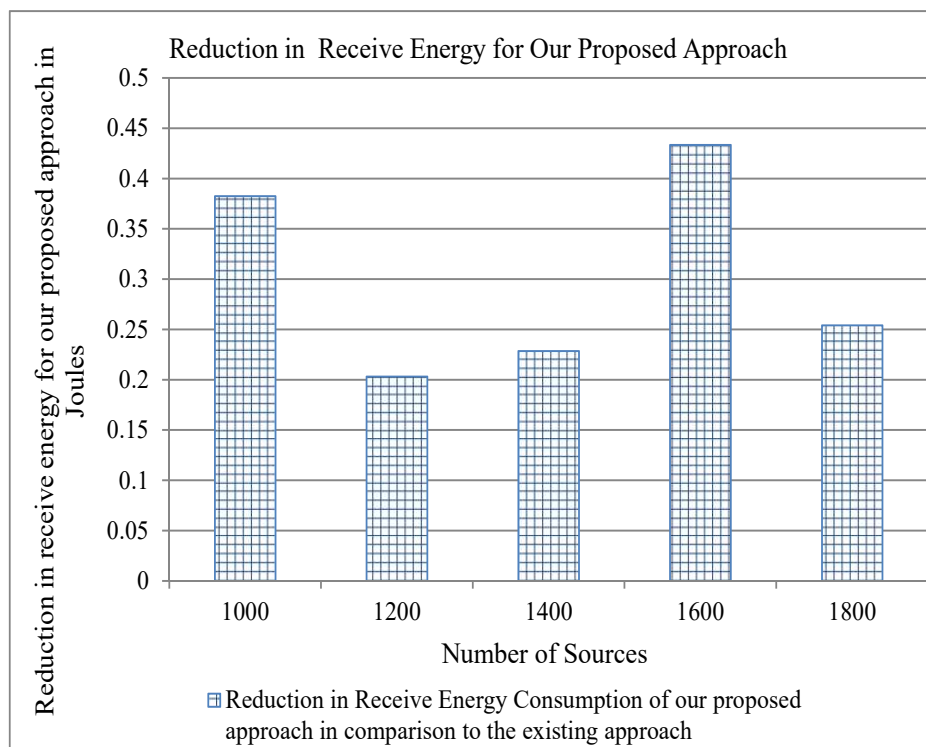


Figure 11. Reduction in Receive Energy for our proposed approach in comparison to the existing for three cache servers.

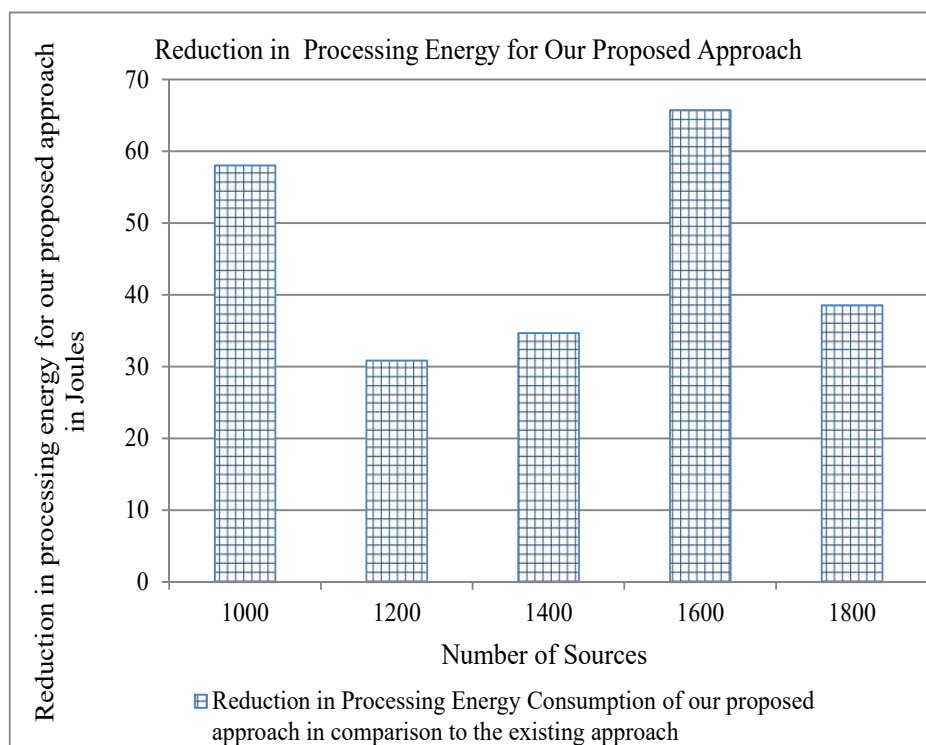


Figure 12. Reduction in Processing Energy for our proposed approach in comparison to the existing for three cache servers.

5. Conclusions

The paper solved the cache servers placement problem for SDN-based ICN using SVD and QR factorization techniques of linear algebra. The proposed approach reduces the computation time from one month to a few seconds as compared to the existing approach. Moreover, the proposed approach also reduces traffic overhead and energy consumption.

Author Contributions: Conceptualization, J.B. and N.S.; methodology, J.B. and N.S.; software, J.B. and N.S.; validation, J.B. and N.S.; formal analysis, J.B. and N.S.; investigation, J.B. and N.S.; resources, J.B. and N.S.; data curation, J.B. and N.S.; writing—original draft preparation, J.B. and N.S.; writing—review and editing, J.B., M.M.A. and N.S.; visualization, J.B. and N.S.; writing—review and editing, supervision, N.S. and M.K.; project administration, J.B. and N.S.; funding acquisition, M.M.A.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TCP/IP	Transmission Control Protocol/Internet Protocol
SDN	Software Defined Network
ICN	Information-Centric Networking
IP Address	Internet Protocol Address
NDN	Name Data Networking
NDO	Name-Data-Object
API	Application Programming Interface
ACL	Access Control List
NFV	Network Functions Virtualization
SAND	Server and Network Assisted DASH
NP	Non-Polynomial
SVD	Singular Value Decomposition
HORST	Home Router Sharing based on Trust
OSN	Online Social Network
ISP	Internet Service Provider
CDN	Content Distribution Network
P2P	Peer-to-Peer
LIFO	Last-In-First-Out

References

1. Cisco Systems. *Cisco Visual Networking Index: Forecast and Methodology*; Technical Report, White Paper; Cisco Systems: San Jose, CA, USA, 2016.
2. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update*; White Paper; Cisco Systems: San Jose, CA, USA, 2017.
3. Kalgoum, A.; Gammar, S.M.; Saidane, L.A. Towards a novel cache replacement strategy for Named Data Networking based on Software Defined Networking. *Comput. Electr. Eng.* **2018**, *66*, 98–113. [[CrossRef](#)]
4. Torres, J.V.; Alvarenga, I.D.; Boutaba, R.; Duarte, O.C.M.B. An autonomous and efficient controller-based routing scheme for networking Named-Data mobility. *Comput. Commun.* **2017**, *103*, 94–103. [[CrossRef](#)]
5. Gao, S.; Zeng, Y.; Luo, H.; Zhang, H. Scalable control plane for intra-domain communication in software defined information centric networking. *Future Gener. Comput. Syst.* **2016**, *56*, 110–120. [[CrossRef](#)]
6. Shah, N.; Giaccone, P.; Rawat, D.B.; Rayes, A.; Zhao, N. Solutions for adopting software defined network in practice. *Int. J. Commun. Syst.* **2019**, *32*, e3990. [[CrossRef](#)]
7. Ioannou, A.; Weber, S. A survey of caching policies and forwarding mechanisms in information-centric networking. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2847–2886. [[CrossRef](#)]
8. Kim, D.; Kim, Y. Enhancing NDN feasibility via dedicated routing and caching. *Comput. Netw.* **2017**, *126*, 218–228. [[CrossRef](#)]

9. Van Adrichem, N.L.; Kuipers, F.A. NDNFlow: Software-defined named data networking. In Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft), London, UK, 13–17 April 2015; pp. 1–5.
10. Seufert, M.; Burger, V.; Wamser, F.; Tran-Gia, P.; Moldovan, C.; Hoßfeld, T. Utilizing home router caches to augment CDNs toward information-centric networking. In Proceedings of the European Conference on Networks and Communications (EuCNC), Paris, France, 29 June–2 July 2015; pp. 1–5.
11. Kalghoum, A.; Saidane, L.A. FCR-NS: A novel caching and forwarding strategy for Named Data Networking based on Software Defined Networking. *Clust. Comput.* **2019**, *22*, 1–14. [[CrossRef](#)]
12. Kalghoum, A.; Gammar, S.M. Towards new information centric networking strategy based on software defined networking. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017.
13. Veltri, L.; Morabito, G.; Salsano, S.; Blefari-Melazzi, N.; Detti, A. Supporting information-centric functionality in software defined networks. In Proceedings of the IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012; pp. 6645–6650.
14. Salsano, S.; Blefari-Melazzi, N.; Detti, A.; Morabito, G.; Veltri, L. Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed. *Comput. Netw.* **2013**, *57*, 3207–3221. [[CrossRef](#)]
15. Mai, H.L.; Aouadj, M.; Doyen, G.; Mallouli, W.; de Oca, E.M.; Festor, O. Toward Content-Oriented Orchestration: SDN and NFV as Enabling Technologies for NDN. In Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA, 8–12 April 2019; pp. 594–598.
16. Aubry, E.; Silverston, T.; Chrisment, I. SRSC: SDN-based routing scheme for CCN. In Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft), London, UK, 13–17 April 2015.
17. Marchal, X.; Cholez, T.; Festor, O. μ NDN: An Orchestrated Microservice Architecture for Named Data Networking. In Proceedings of the ACM-ICN'18-5th ACM Conference on Information-Centric Networking, Boston, MA, USA, 21–23 September 2018.
18. Zhang, Q.Y.; Wang, X.W.; Huang, M.; Li, K.Q.; Das, S.K. Software defined networking meets information centric networking: A survey. *IEEE Access* **2018**, *6*, 39547–39563. [[CrossRef](#)]
19. Lai, J.; Fu, Q.; Moors, T. Using SDN and NFV to enhance request rerouting in ISP-CDN collaborations. *Comput. Netw.* **2017**, *7*, 176–187. [[CrossRef](#)]
20. Arumaithurai, M.; Chen, J.; Maiti, E.; Fu, X.; Ramakrishnan, K.K. Prototype of an ICN based approach for flexible service chaining in SDN. In Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOMWKSHPS), Hong Kong, China, 26 April–1 May 2015.
21. Petropoulos, G.; Katsaros, K.V.; Xezonaki, M.E. OpenFlow-compliant topology management for SDN-enabled Information Centric Networks. In Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Heraklion, Crete, Greece, 3–6 July 2017.
22. Nascimento, E.B.; Moreno, E.D.; de Macedo, D.D.J. A Programmable Network Architecture for Information Centric Network using Data Replication in Private Clouds. In Proceedings of the IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Poznan, Poland, 21–23 June 2017; pp. 137–142.
23. Mahmood, A.; Casetti, C.; Chiasserini, C.F.; Giaccone, P.; Härrä, J. Efficient caching through stateful SDN in named data networking. *Trans. Emerg. Telecommun. Technol.* **2018**, *29*, 3271. [[CrossRef](#)]
24. Ghosh, U.; Chatterjee, P.; Tosh, D.; Shetty, S.; Xiong, K.; Kamhoua, C. An SDN based framework for guaranteeing security and performance in information-centric cloud networks. In Proceedings of the IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, CA, USA, 25–30 June 2017; pp. 749–752.
25. Liu, Z.; Zhu, J.; Zhang, J.; Liu, Q. Routing algorithm design of satellite network architecture based on SDN and ICN. *Int. J. Satell. Commun. Netw.* **2019**, *38*, 1–15. [[CrossRef](#)]
26. Xing, C.; Ding, K.; Hu, C.; Chen, M.; Xu, B. SD-ICN: Toward Wide Area Deployable Software Defined Information Centric Networking. *KSII Trans. Internet Inf. Syst.* **2016**, *10*, 2267–2285.
27. Yu, M.; Rexford, J.; Freedman, M.J.; Wang, J. Scalable flow-based networking with DIFANE. *ACM Sigcomm Comput. Commun. Rev.* **2011**, *41*, 351–362.

28. Badshah, J.; Kamran, M.; Shah, N.; Abid, S.A. An Improved Method to Deploy Cache Servers in Software Defined Network-based Information Centric Networking for Big Data. *J. Grid Comput.* **2019**, *17*, 255–277. [[CrossRef](#)]
29. Clayman, S.; Kalan, R.S.; Sayit, M. Virtualized Cache Placement in an SDN/NFV Assisted SAND Architecture. In Proceedings of the IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Batumi, Georgia, 4–7 June 2018.
30. Hohlfeld, O.; Kempf, J.; Reisslein, M.; Schmid, S.; Shah, N. Guest editorial scalability issues and solutions for software defined networks. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2595–2602. [[CrossRef](#)]
31. Golub, G.H.; Van L.C.F. Matrix computations. In *Matrix Computations*; JHU Press: Baltimore, MD, USA, 2007; pp. 32–58.
32. Seufert, M.; Burger, V.; Hoßfeld, T. HORST-Home router sharing based on trust. In Proceedings of the IEEE 9th International Conference on Network and Service Management (CNSM), Zurich, Switzerland, 14–18 October 2013; pp. 402–405.
33. Orgerie, A.C.; Amersho, B.L.; Haudebourg, T.; Quinson, M.; Rifai, M.; Pacheco, D.L.; Lefèvre, L. November. Simulation toolbox for studying energy consumption in wired networks. In Proceedings of the 13th IEEE International Conference on Network and Service Management (CNSM), Tokyo, Japan, 26–30 November 2017.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).