sensors

MDPI

Article

Caching Joint Shortcut Routing to Improve Quality of Service for Information-Centric Networking

Baixiang Huang [1], Anfeng Liu [1,2], Chengyuan Zhang [1,*], Naixue Xiong [3], Zhiwen Zeng [1] and Zhiping Cai [4]

1 School of Information Science and Engineering, Central South University, Changsha 410083, China; bxhuang@csu.edu.cn (B.H.); afengliu@mail.csu.edu.cn (A.L.); zengzhiwen@mail.csu.edu.cn (Z.Z.)
2 The State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China
3 Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, OK 74464, USA; xiongnaixue@gmail.com
4 Department of Network Engineering, School of Computer, National University of Defense Technology, Changsha 410073, China; zpcai@nudt.edu.cn
* Correspondence: cyzhang@csu.edu.cn; Tel.: +86-731-8887-9628

Received: 7 March 2018; Accepted: 24 May 2018; Published: 29 May 2018

Abstract: Hundreds of thousands of ubiquitous sensing (US) devices have provided an enormous number of data for Information-Centric Networking (ICN), which is an emerging network architecture that has the potential to solve a great variety of issues faced by the traditional network. A Caching Joint Shortcut Routing (CJSR) scheme is proposed in this paper to improve the Quality of service (QoS) for ICN. The CJSR scheme mainly has two innovations which are different from other in-network caching schemes: (1) Two routing shortcuts are set up to reduce the length of routing paths. Because of some inconvenient transmission processes, the routing paths of previous schemes are prolonged, and users can only request data from Data Centers (DCs) until the data have been uploaded from Data Producers (DPs) to DCs. Hence, the first kind of shortcut is built from DPs to users directly. This shortcut could release the burden of whole network and reduce delay. Moreover, in the second shortcut routing method, a Content Router (CR) which could yield shorter length of uploading routing path from DPs to DCs is chosen, and then data packets are uploaded through this chosen CR. In this method, the uploading path shares some segments with the pre-caching path, thus the overall length of routing paths is reduced. (2) The second innovation of the CJSR scheme is that a cooperative pre-caching mechanism is proposed so that QoS could have a further increase. Besides being used in downloading routing, the pre-caching mechanism can also be used when data packets are uploaded towards DCs. Combining uploading and downloading pre-caching, the cooperative pre-caching mechanism exhibits high performance in different situations. Furthermore, to address the scarcity of storage size, an algorithm that could make use of storage from idle CRs is proposed. After comparing the proposed scheme with five existing schemes via simulations, experiments results reveal that the CJSR scheme could reduce the total number of processed interest packets by 54.8%, enhance the cache hits of each CR and reduce the number of total hop counts by 51.6% and cut down the length of routing path for users to obtain their interested data by 28.6–85.7% compared with the traditional NDN scheme. Moreover, the length of uploading routing path could be decreased by 8.3–33.3%.

Keywords: Information-Centric Networking; routing shortcut; cooperative pre-caching; Quality of Service

1. Introduction

Rapid advances in the manufacture of sensing devices such as smartphones [1–4], iPad or other sensing nodes [5–9] have expanded the range of ubiquitous sensing applications in

the Internet-of-Things (IoT) [3,9–13], such as monitoring and gathering information from public infrastructure [14–18], natural disaster relief [19,20], healthcare [21,22], smart homes [23,24], and industries [11,16,17,25,26]. The enormous number of ubiquitous sensing (US) devices provides an immense number of data (in this paper, the term "data" refers to videos, sounds, pictures or other forms of information that could be collected by sensing devices) for Information-Centric Networking (ICN) [27], which is an emerging network architecture with the potential to solve a great variety of issues faced by traditional network [27]. In addition, those applications formed the so-called Sensor-Cloud Network (SCN) [28], which is an emerging network architecture of ICN [27]. In such a network, many sensing devices which are deployed on the network edge are used for collecting data [2,4,29–33]. Then, sensing devices send data to cloud [28,34], where Data Centers (DCs) are located. Finally, users get requested data from DCs. Ubiquitous sensing devices, including smartphones, cameras and iPads have been changing the ways we interact and communicate dramatically, which lead to remarkable changes in the existing network [35–39]. On the one hand, it is the enormous number of these ubiquitous sensing devices that make it possible for data to be collected and sensed in such a huge scale [25]. According to the statistics [1,3], the total number of sensing devices (smartphones and industrial sensing devices) that are connected to the IoT network had reached 90 billion in 2011, which outnumbered the global population by that year. Moreover, this number is expected to ascend by 240 billion in the immediate future [1,3]. On the other hand, the growth of streaming traffic has also shown an exponential trend under pervasive sensing [1,3]. Due to these powerful sensing devices, people could sense videos, sounds and photos anytime and connect their devices to the Internet using various access channels. The convenience brought by sensing devices make it possible for hot events, regional conflicts and political issues to be spread worldwide almost simultaneously. Because of the colossal number of sensing devices around the world, the number of data produced by the IoT has been rising exponentially [1,3,40,41]. Reports from Cisco illustrate that the network flow of streaming produced by the IoT had accounted for 69% of the total flow in 2014, which had increased to 30-fold of the total network flow in 2000. This number is expected to grow more dramatically in the future [1,3].

The conventional Cloud Computing model [1,25,28,34] and the Sensor-Cloud Network [28,34] have to face considerable challenges brought by the rapid development of ubiquitous sensing [42–44]. In a traditional model, all the sensed data collected by sensing devices have to be uploaded to DCs which are located in the cloud, then it is DCs' job to reply every incoming interest packet with a data packet with a specified size and name same as in that interest packet. Nevertheless, this model has a few disadvantages: (1) It causes heavy traffic load for DCs and backbone network because the network center where DCs are located is the core of data computing and downloading [45,46]. On the one hand, DCs have the burden of computation and data downloading. On the other hand, tons of streaming data collected from sensing devices are uploaded to DCs, which results in significant traffic flow between DCs and backbone network. (2) Traditional model can lead to poor Quality of Service (QoS) for users [47]. To obtain the first-hand information such as emergency events, public hot issues, or interesting news, users have to go through some prolonged routing processes. Firstly, newly sensed data are uploaded towards DCs; secondly, users request their interested data from DCs; and, finally, demanded data would be transferred from DCs to users. Therefore, these interacting processes result in long delay for a user to get requested data and aggravate the burden of network, thereafter it is the time lag of data transmission that causes poor QoS. Nowadays, located on the edge of the network, many devices have high storage capacity but their potential has not been fully exploited. State-of-art schemes such as Fog Computing [48] and Edge Computing network [24,45] are trying to make use of the resources on the edge of network. However, if the fundamental processes remain unchanged, the abundant computation and storage resources cannot be fully exploited.

Some researchers noticed the problems faced by the Sensor-Cloud Network. Among various proposed schemes, caching is widely used to address many of existing issues [24,35]. In those caching schemes, the major idea is that, when a user requests data, the requested data would be cached in routers along the transmission path of that data. When other users request the same data, they can

directly fetch the data from routers, and the traffic burden of delivering data would hence be released and QoS would be ameliorated [35,47].

Nevertheless, the majority of past research mainly concentrates on downloading routing from DCs to users. However, besides the downloading process between users and DCs, there are a few other routing processes in SCN. In the initial place, all data are collected from sensing devices on the network edge, and users could get access to their interested data after the data have been uploaded to DCs. Therefore, there are three main processes from when data are produced to when the data are obtained by users: (1) Sensing devices collect and upload data to DCs. (2) Users request specific data from DCs. (3) DCs send the requested data towards users. These three processes are intimately correlated; hence, the distribution and transmission of data would be influenced if any of them is ineffective. The goals of previous caching schemes are to reduce data traffic or delay. Although those works are mainly based on the third process and pay little attention to the remaining processes. In this paper, we believed that the other two processes are also critical and if all of the processes are considered and optimized together, the network performance could be further enhanced. Thus, we proposed a Caching Joint Shortcut Routing (CJSR) scheme to optimize the overall performance of the network. Compared with other works, the major innovations of this work are as follows:

(1) Two shortcut routing methods are proposed to reduce the burden of network and enhance uses' QoS. The first shortcut routing method is to build shortcut from DPs to users. In the SCN, the sensed data collected from sensing devices has to be uploaded to DCs before it can be requested by users later. However, sensing devices and users are both located on the edge of network, which means that sensing devices (DPs) and users are close to each other. Making use of this advantage in distance, the data could be directly sent to users after the formation of the shortcut. The second shortcut routing method will choose a Content Router (CR) that could yield shorter overall length of uploading routing paths, and then data packets are uploaded through this chosen CR. In this method, the uploading path shares some segments with the pre-caching path thus the overall length of routing paths is reduced. By implementing these shortcut routing methods, the latency for users to get their interested data would decrease and a better QoS could be guaranteed.

(2) A cooperative pre-caching mechanism consisting of uploading pre-caching and downloading pre-caching is proposed so that the efficiency of pre-caching could be improved further. In the CJSR scheme, pre-caching is not only performed between DCs and users similar to past schemes, but data are also cached providently during their uploading process. In the uploading pre-caching, before data have been uploaded to DCs, they would be cached in CRs by the time users request then, which would result in better QoS because users could fetch data from nearby CRs. Besides the proposed uploading pre-caching, the CJSR scheme also provides pre-caching for downloading process so that it is more adaptive to different situations and could enhance the system performance. Moreover, this paper also proposes a space-optimize algorithm that could make use of storage of idle CRs to address the scarcity of storage capacity. Overall, the CJSR scheme changes the conventional one-way model of pre-caching and enhance the network performance with a broader perspective.

(3) The effectiveness of the CJSR scheme is evaluated through extensive simulations. Because of the two kinds of shortcuts, the CJSR scheme could reduce the overall length of routing paths, decrease delay, release transmission burden and ameliorate user's QoS. Moreover, with cooperative pre-caching, the caching feature of ICN could be exploited so that the network traffic burden is decreased further. In addition, to utilize the caching storage of each CR in a balanced way, another pre-caching scheme that could make use of idle CRs is proposed, which could relieve the scarcity of storage space. After conducting a series of experiments, we demonstrate that, compared with the traditional NDN scheme, the CJSR scheme could reduce the total number of processed interest packets by 54.8%, enhance the cache hits of each CR and reduce the number of total hop counts by 51.6% and cut down the length of routing path for users to obtain their interested data by 28.6–85.7%. Moreover, the length of uploading routing path could be decreased by 8.3–33.3%.

The rest of this paper is organized as follows: in Section 2, related works are reviewed. The system model is described in Section 3. In Section 4, a novel Caching Joint Shortcut Routing (CJSR) scheme is presented. Section 5 presents our experimental results and comparisons with literature methods. Finally, a conclusion and future works are presented in Section 6.

2. Related Work

Along with far-reaching development of the Internet, the traditional TCP/IP network has shown many defects facing the rapid growth of network scale as well as an ever-increasing number of data. These problems include low network performance, poor QoS and inefficient data transmission among DPs, DCs and users. Therefore, researchers carried out many relevant studies to improve data caching mechanisms, architectures of network and routing methods [27,49,50]. As a future network architecture, Information-Centric Networking (ICN) [27,51] has brought much attention from both academia and industry. In ICN, named data become a major unit of network transmission, Named Data Networking (NDN) [27,51] is a paradigm of ICN architecture. In NDN, the IP in middle layer is replaced by the named data. The transmission of data follows a "publish-request-respond" model which directly uses the named data in routing, thus the efficiency from point to multi-point transmission is improved.

Majeed et al. [35] and Chiocchetti et al. [51] indicated that caching strategy is a crucial factor to enhance network performance for NDN. In the initial place, past research proposed a Cache Everything Everywhere (CE2) scheme, of which the cached contents (the words "content", "data packet" and "Data" are used interchangeably in this paper) tend to be homogenized and thus result in a large amount of redundancy. Due to the lack of consideration on the heterogeneous feature of contents, efficient caching of contents cannot be achieved in the CE2 scheme. Therefore, to exploit the potential of caching, designing an effective caching algorithm becomes a prime goal for NDN.

Many researchers brought up various caching methods. The earliest caching scheme of NDN is LCE (Leave Copy Everywhere) Caching Scheme, in which an interest packet finds outgoing faces by looking up entries in the Forwarding Information Base (FIB), then an entry about the request content is created in the Pending Interest Table (PIT). The corresponding data packet will be sent back along the same path that the interest packet had travelled, and then it is cached by each CR along this path [52]. However, the storage capacity of CRs is limited and copying same content everywhere causes excessive redundancy. Caching mechanism in the LCE would not only waste valuable storage space, but also render low efficiency due to the frequent replacement of contents in CRs. Therefore, the LCE caching scheme is easy to deploy but inefficient.

Earlier, Laoutaris et al. proposed the LCD (Leave Copy Down) [17], MCD (Move Copy Down) [53] and Prob (copy with probability) [53] schemes. In the LCD scheme, data are only cached in next vertex of the vertex where a cache hit happens. In the MCD scheme, when a cache hit happens, the same data would be deleted in this vertex, and then those data are cached in next vertex, similar to the LCD scheme. In the Prob scheme, each CR caches a data packet based on a predefined probability p ($0 < p < 1$). These three schemes can reduce the redundancy of cached data in the whole network to some extent. In addition, the LCD and the MCD scheme also consider the frequency of requested data. When data are requested frequently, they would be cached closer to users. Nevertheless, in these two schemes, the vertices that are close to servers have higher burden than the other vertices and the potential of those underused vertices are not fully exploited.

Chai et al. [54] proposed the Betw scheme, in which the importance of a vertex is judged by calculating its betweenness (higher betweenness denotes higher importance). In the process of forwarding data to users, the data would be cached in the vertex with the highest betweenness. In this scheme, each vertex has to compute its own betweenness; higher betweenness means that this vertex is more significant, thus caching data in such a vertex render a higher probability for users to get their interested data, and the network performance is enhanced. However, the Betw scheme would put higher caching pressure on vertices with high betweenness, and the storage capacity of

low-betweenness vertices is not efficiently used. Moreover, data replacement in high-betweenness vertices is frequent, which means popular data are also likely to be replaced very often.

Psaras and Chai et al. [55] proposed the Prob Cache scheme. In this scheme, after sending an interest packet, a probability value would be calculated using the storage capacity of a CR on the path that the interest packet had just travelled and the distance between this CR and users. Then, this number would be referred as a criterion on caching data packets. The Prob scheme is also used widely by later research. Overall, it is a scheme that, after evaluating the importance of each vertex, cache data according to the importance of vertices. This scheme reduces redundancy of data packets to some extent, improves efficiency on using storage capacity and ameliorate the network performance. However, popularity of data is not considered in this scheme, and all the data are treated equally. As a result, the popular contents are cached inefficiently.

Ming et al. [56] proposed the age of cache, which is calculated by using popularity of a content and the position of the vertex that stores this content. This work also combined age with other factors to enhance the performance of schemes that simply replace the old content with a new one. Cho et al. [57] proposed a collaborative WAVE scheme that based on the popularity of content packets, the number of cache on a content packet grows exponentially according to the number of request on this content. In addition, Zhang et al. [58] made a comparison on existing caching schemes, indicating that there was a lack of consideration on the distribution attributes of content requests in the design of those caching algorithms. They also gave an inspiration on possible future research directions.

Cache-based technologies is of great importance for NDN, however the development of network has risen new challenges for these technologies. In fact, existing caching schemes assume that data have always been uploaded to DCs by the time it is requested by a user, and the above-discussed schemes are all based on this assumption. However, such an assumption is only applicable for network ten years ago; the concept of = Cloud Computing has since been proposed. The core idea of the Cloud Computing is to deploy hardware resources that have powerful computation ability and high storage capacity in network center; it is similar to the structure of fat server that can undertake a large number of complicated computation and storage. On the other hand, a terminal is rather simple, which gets the result of computation after sending request to the cloud. In this way, the model of Cloud Computing hides the physical difference of devices and provide distributed services. Nevertheless, with the development of IoT, the number of devices that are connected to the Internet grows substantially [1,3,59]; the scale and range of producing data and sharing data also increase exponentially. Because all the data produced by numerous devices must be sent to network center (Cloud platforms or DCs) for computing and processing, the overall burden of the network center is significant, delay is high and QoS is poor. On the one hand, devices in the network center are overloaded and network is often congested. On the other hand, computation and storage resources of many devices on the edge are underused. Thus, computational frameworks such as Fog Computing and Edge Computing are proposed. Their main idea is to shift the work of computing from network center to edge to exploit the potential of computation and storage resources. Under that model, users' request would be satisfied in a local network. The congestion and latency caused by long-distance transmission between users and DCs could be decreased. However, the research of distributed network architectures is still in early stage of exploration.

Because the emerging network architectures are immature, the original network architecture is still widely used, but its caching mechanisms are inefficient. Specifically, those caching schemes are only applicable during routing between DCs and users. In other words, it could only solve partial problem of down data flow from network center to network edge. Performance of the process of uploading immense data from sensing devices to DCs has not been improved by past schemes, not to mention, more broadly, trying to optimize those processes together. Nowadays, main pressure on the network is caused by the up data flow, or the data flow from edge to edge, which has to be relayed by network center (for instance, a colossal number of data produced by instant messaging such as QQ and WeChat must be relayed by network center). In conclusion, the CJSR scheme in this paper could

be a possible solution based on such a background; this scheme considers the attributes of present network and aims to solve the existing problems.

3. System Model and Problem Statement

3.1. Network Model

As shown in Figure 1, the network model used in this paper is Sensor-Cloud Network (SCN), which is an emerging network model in ICN. In such a network, there are three major components:

(1) Ubiquitous sensing devices. Broadly distributed sensing devices are producers of data. The information foundation of ICN is based on a colossal number of data collected and sensed by these devices and the number of these devices are numerous. Located on the network edge, they are important parts of the Internet of Things (IoT). These sensing devices are made up of mobile sensing devices such as portable devices (smartphones, iPad and cameras) and more powerful devices carried by vehicles, ships and unmanned aerial vehicles. There are also static sensing devices including surveillance cameras and industrial sensors. Data can be sensed in formats such as video, audio, image or text. For instance, emergency events or breaking news are usually collected by sensing devices of nearby people in forms of videos, audios and images. Following that, the collected content is uploaded to DCs for storage. Ubiquitous sensing devices are shown in the left part of Figure 1.

(2) Data Centers (DCs). DCs are similar to publishers in Named Data Networking (NDN), which provide all the requested data to users and have the most comprehensive storage of all kinds of data. Rather than focus on the transmission of stored video streaming similar to other research on NDN, the data of SCN in this paper have semi-real-time attribute. When an emergency event or breaking news happens, users want to obtain relevant videos, audios and images as soon as possible. Lagged data transmission would result in loss of interest; these data also become out-of-date as time goes by. For instance, during a traffic congestion, drivers in a congestion area send the current traffic situation to a DC, other users could get this information and use it to schedule their routes. If the obtained data are outdated, users (especially those who plan to drive towards this road) would make wrong decisions. One possible situation is that the outdated information may lead users to drive through a longer road because they thought the shorter road is congested. As a result, this longer road would cost more time and gasoline. Users could also be misled to a congested road by outdated information which reports that this road is unobstructed. Overall, this information would give users a poor QoS. Besides the example we give above, similar problems also happen on other applications and could cause users to make wrong decisions. In this paper, DCs are different from those in traditional ICN, in which it is assumed that all data have been uploaded whenever users request them. Traditionally, data are usually obtained by physical methods such as installing storage hardware (hard disks, CDs or tape recorders). Those data are mainly multi-media streaming with low real-time request. Thus, data producers in traditional ICN are usually omitted and only the interaction process between DCs (publishers) and users is considered. However, in this paper, we aim to improve performance on processing semi-real-time data, and have considered a situation that the requested data have not been uploaded completely. Besides stored videos used in traditional ICN, we focus on real-time videos collected from sensing devices around locations where breaking news occurred, and collected video streaming is often short (approximately lasts between a few seconds and ten minutes). Such real-time data would provide users better QoS if they could be delivered more swiftly; vice versa, outdated data have less value and QoS would be declined under their influence.

(3) Users. In this paper, users are equivalent to data requesters, such as drivers in the aforementioned example, or people who are interested in breaking news or a hot event. Due to the powerful functions of smartphones, more and more users obtain data through smartphones or iPad rather than personal computers. Thus, smartphones are not only producers (Data Producers of DCs) which sense and collect data, but also data consumers. It is also noticeable that users and DPs (sensing devices) are both located on the network edge.

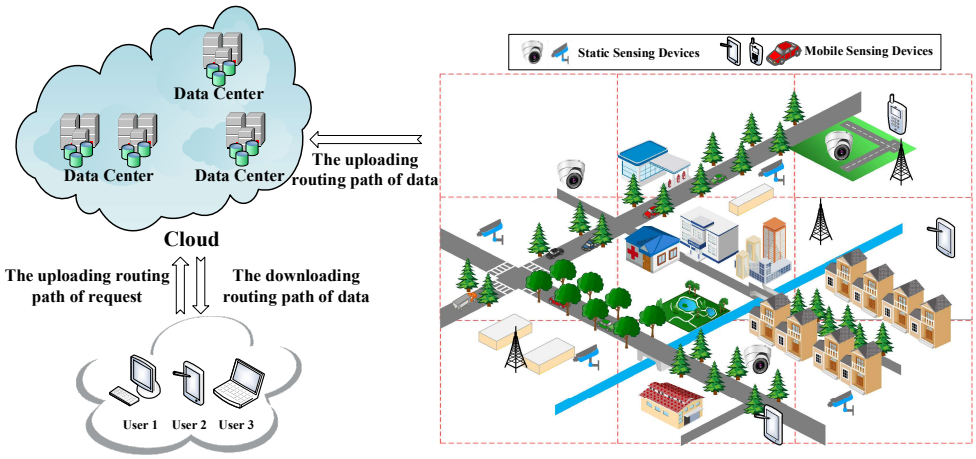

Figure 1. The architecture of the network.

Besides the three components discussed above, the network model contains the following interaction processes:

(1) Routing process of requesting data from DCs. Although users and DPs are both located on the network edge, prefixes of sensing devices are unknown to users, but prefixes of DCs are public to the whole network. Therefore, a user would firstly request data from DCs as they usually do in conventional SCN. This routing path is built between users and cloud; we refer to it as routing path of request. To build the first kind of shortcut that enables users to directly send requests to sensing devices rather than DCs, a DP's prefix would be recorded as soon as a DC receives uploaded data. Hence, with the prefix recorded, the DC could reply the information of the DP to the user if the requested data have not been uploaded yet. Then, the user could directly request data from the DP. Otherwise, the DC would send corresponding data to the user following the traditional procedures.

(2) Routing process of transmitting data from DCs to users. When DCs receive requests from users, they send corresponding data to users if the requested data are stored. This routing path is the transmission path of data between DCs and users, thus this path is referred as downloading routing path of data.

(3) Routing process of uploading data. While sensing devices are collecting data, they upload collected data to DCs at the same time without being requested by DCs. The data collection process in the network model which our work used is similar to crowdsourcing network or participatory network [14–19]. Therefore, data packets are uploaded without being requested by the cloud. The routing path in this process is referred to as the routing path of uploading data. This routing path is usually directly built among DCs and DPs. However, using two shortcut routing methods discussed in Section 4, this path would adjust itself to yield shorter overall routing paths.

Next, we give standardized definition of the network model shown in Figure 1. The network can be defined as an undirected graph $G = (\xi, E)$, where ξ is a set of vertices or nodes representing content routers, data requesters (DRs), data producers (DPs) and the data centers (DCs). Usually, DRs, DPs and DCs are located on the terminal of a routing path while CRs are intermediary. E is the set of undirected edges modeling communication links between pairs of vertices.

(1) Suppose that $\xi = \{\xi_{i1}, \xi_{i2}, \ldots, \xi_{in}\}$ is a set of intermediary content routers (CRs) on a routing path (start with a requester and end at a DC or a DP). The CRs can cache the data in their cache storage. A vertex ξ_i has a limited cache storage capacity C_{ξ_i}, measured in bytes.

(2) Suppose that $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$ is the set of data requesters (DRs), i.e., users. The data requesters request data and if the data have been cached among CRs, the hop count for users to obtain these data would be reduced. Users could obtain data from a CR with these data cached, thus the QoS is ameliorated. Similar to a previous research [35], we assume that data requesters are connected to network through a single CR. We denote the vertex (CR) connected to requester r_i as ξ_{r_i}.

(3) Suppose that $\mathfrak{S} = \{p_1, p_2, \ldots, p_k\}$ is the set of data producers (DPs). The data producers are sensing devices that sense and collect data. Sensing devices provide data to users or DCs through CRs. Different with other research, data can also be cached during the transmission from DPs to DRs in our work. We denote the vertex (CR) connected to data producer $\mathcal{S}_i$ as ξ_{p_i}.

(4) Suppose that $\mathfrak{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_\mathcal{N}\}$ is the set of data centers (DCs). Data Centers have powerful storage and computation capacity and their addresses are public. Besides the data that have not been uploaded, they have stored all the data that may be requested by users. We denote the vertex (CR) connected to data center $\mathcal{D}_i$ as $\xi_{\mathcal{D}_i}$.

(5) Let $S_n = \{S_1, S_2, \ldots, S_m, \ldots, S_n\}$ be a set of data packets. For a requested chunk S_m from S_n, we define $\mathcal{L}$ as the fixed size of each data packet. $\mathcal{A}^m_{\xi_i, r_j}$ denotes the frequency of interest packets for S_m at node $\xi_i \in V$ from user r_j.

(6) Suppose that $\mathbb{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_i, \ldots, \mathcal{P}_\mathcal{M}\}$ is a set of routing path. $\mathcal{P}_i = \{\xi_{i1}, \xi_{i2}, \ldots, \xi_{in}\}$, in which ξ_{i1} is the first vertex on this routing path. There are a few possible scenarios: (a) When users start to request data through the uploading routing path, ξ_{i1} is a user (ξ_{r_i}), while ξ_{in} could be a CR, a DC or a DP, if the request is satisfied by a CR in the routing path, ξ_{in} is that CR; if the request is satisfied by a DC, ξ_{in} is that DC ($\xi_{\mathcal{D}_i}$); if the requested data have not been uploaded yet, users will request from a DP, in that case, ξ_{in} is a DP (ξ_{p_i}). To facilitate pre-caching, we make use of the transmission path of an interest packet from user to DCs or DPs, and we refer to this path as $\mathcal{P}_{cache}$. $\mathcal{P}_{cache} = \{\xi_{i1}(\xi_{r_i}), \xi_{i2}, \ldots, \xi_{ij}(\xi_{p_i}\ or\ \xi_{\mathcal{D}_i})\}$. We discuss the details of its construction in Section 4.1. (b) During the transmission of data, ξ_{i1} could be a CR or a DC while ξ_{in} is the user who issues this request (ξ_{r_i}). (c) For the uploading routing path of data, ξ_{i1} is a DP, and ξ_{in} could be a user, CR or DC. If ξ_{in} is a user, it denotes that data are forwarded from a DP to a user; if ξ_{in} is a CR, this routing path starts from a DP to a CR; and, if ξ_{in} is a DC, it is a between a DP and a DC. Thus, the first and the last vertex on routing path $\mathcal{P}_i = \{\xi_{i1}, \xi_{i2}, \ldots, \xi_{in}\}$ is usually ξ_{r_i}, $\xi_{\mathcal{D}_i}$ or ξ_{p_i}.

For the convenience of readers, Table 1 summarizes the notations used in this paper.

Table 1. Notations.

Notation	Description
CR	Content Router
DC	Data Center
DP	Data Producer
S_n	A data packet
I_n	An interest packet for S_n
$G = (\xi, E)$	An undirected graph, ξ and E denote vertex and edge respectively
$\mathcal{P}_{cache}$	The routing path used for pre-caching
ξ_{ij}	An intermediate vertex on the $\mathcal{P}_{cache}$
ξ_{p_i}	A vertex on the $\mathcal{P}_{cache}$ adjacent to a Data Producer P_i.
ξ_{r_i}	A vertex on the $\mathcal{P}_{cache}$ adjacent to a user r_i
$\xi_{\mathcal{D}_i}$	A vertex on the $\mathcal{P}_{cache}$ adjacent to a Data Center $\mathcal{D}_i$
C_{ξ_i}	The storage capacity of ξ_i
N_ξ	The number of CRs along the $\mathcal{P}_{cache}$
P_m	The sequence number of a CR in the PathMark element
$\mathcal{A}^m_{\xi_i, r_j}$	The Interest frequency of I_n on ξ_{ij} from user r_j

3.2. Problem Statement

The optimization objectives of the CJSR scheme are different from past schemes that merely focus on performing pre-caching from DCs to users. For those past schemes, their objective is to minimize the total number of hop counts for users to obtain data. This paper gives a more comprehensive consideration by taking into account of three interaction processes that we discuss in Section 3.1. Hence, the CJSR scheme is more practical for SCN. The optimization objectives in this paper include:

(1) Minimization of the total hop counts for users to obtain data. Assume that the path for users to obtain data is the same as previous definitions: $\mathbb{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_i, \ldots, \mathcal{P}_\mathcal{M}\}$ and $\mathcal{P}_i = \{\xi_{i1}, \xi_{i2}, \ldots, \xi_{in}\}$. Then, the length of the ith routing path is denoted as $\mathcal{H}_i = |\mathcal{P}_i|$. The first

objective in this paper is to minimize the hop counts that users obtain their interested data, as demonstrated by Equation (1).

$$\min(\mathbb{H}) = \min\left(\sum_{1\leq i\leq \mathcal{M}} |\mathcal{P}_i|\right) \tag{1}$$

(2) Minimization of delay. The optimization objective of past schemes is similar to Equation (1). Besides, we have more objectives in this paper. The delay refers to the whole time interval from when data are requested to when they are finally received by users, and reflects the timeliness for a user to receive data. Obviously, less delay would lead to better QoS. Assume the time that a data chunk $C_{i,j}$ is produced is $\mathcal{G}_{i,j}$, the time when it is received by user k is $\mathcal{J}^k_{i,j}$. Thus, the delay of user k requests for data chunk $C_{i,j}$ is $\Gamma^k_{i,j} = \mathcal{J}^k_{i,j} - \mathcal{G}_{i,j}$. Thus, the second optimization objective is Equation (2)

$$\min(\Gamma) = \min\left(\sum_{1\leq i\leq m, 1\leq j\leq \gg}^{1\leq k\leq m} \Gamma^k_{i,j}\right) \tag{2}$$

(3) Minimization of network burden. The burden of network could be measured by the number of packets forwarded by each CR. Reduce the burden is also one of the CJSR scheme's objectives. It is assumed that the number of packets forwarded by a CR ξ_i is $\mathfrak{S}_i$. The third optimization objective in this paper is Equation (3).

$$\min(\mathbb{O}) = \min\left(\sum_{1\leq i\leq m} \mathfrak{S}_i\right) \tag{3}$$

Obviously, the goal of the CJSR scheme is to minimize the total number of hops $\mathbb{H}$, minimize delay Γ and minimize network burden $\mathbb{O}$. A constraint for achieving these optimization goals is: The caching storage of a CR ξ_i at some time ($\mathcal{C}_{\xi_i}$) cannot exceed its physical storage capacity C_{ξ_i}. The optimization goals in this paper can be summarized as follows.

$$\begin{cases} \min(\mathbb{H}) = \min\left(\sum_{1\leq i\leq \mathcal{M}} |\mathcal{P}_i|\right) \\ \min(\Gamma) = \min\left(\sum_{1\leq i\leq m, 1\leq j\leq \gg}^{1\leq k\leq m} \Gamma^k_{i,j}\right) \\ \min(\mathbb{O}) = \min\left(\sum_{1\leq i\leq m} \mathfrak{S}_i\right) \\ s.t.\ \forall i \mid \mathcal{C}_{\xi_i} \leq C_{\xi_i} \end{cases} \tag{4}$$

4. Main Design of the CJSR Scheme

4.1. The Design of Data Structures

In this section, we discuss data structures related to the CJSR scheme. Our scheme is an improvement based on previous in-network caching schemes and we aim to improve caching mechanisms and reduce the length of routing paths. Firstly, three classic data structures including the Content Store (CS), the Pending Interest Table (PIT) and the Forwarding Information Base (FIB) are used and the lookup procedures are depicted in Figure 2. Other data structures used in the CJSR scheme include: (1) interest packet, which is used by users to request data and to construct a fixed path to facilitate pre-caching; (2) data packet, the function of which is to carry data and relevant information about data (like its destination); (3) Extra Storage (ES), which is a table sharing a similar structure to the Content Store. The ES is used in Algorithm 3 for an idle CR to help store data packets.

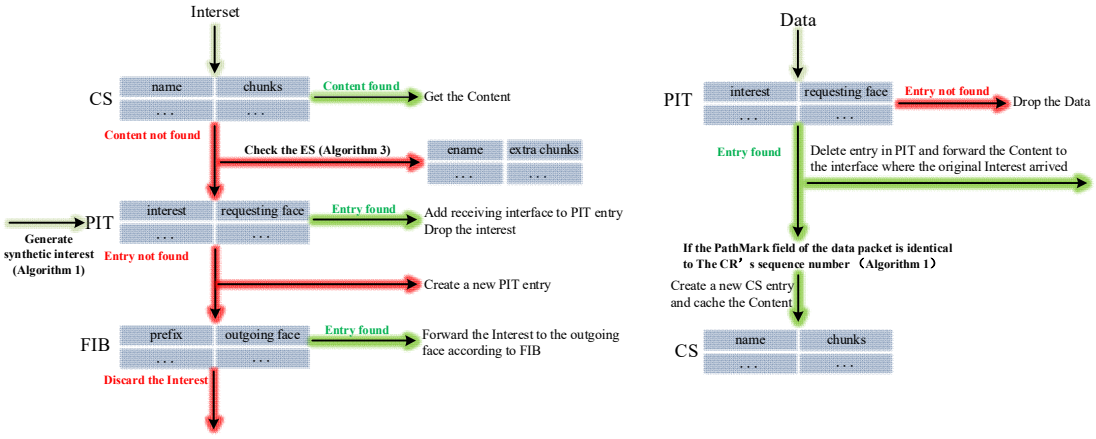

Figure 2. Lookup procedures of the CJSR scheme.

(1) Interest packet format. The structure of an interest packet is demonstrated in Figure 3 (the tilde marks represent that the length of this field is variable). Besides Name and Nonce, Selectors are optional elements that further qualify Data that may match the Interest. They are used for discovering and selecting the data that match best to what the application wants, and this is the place where the PathMark element is added. The aim of adding this element is to record a series of CRs that this interest packet had travelled. Thus, in the CJSR scheme, with the added PathMark, a DC or a DP can construct a fixed path to facilitate pre-caching. The reason for constructing such a path is to providently assign data packets into CRs in a controllable way. Traditionally, data packets would be sent back through the transmission path of interest packets. Therefore, it is reasonable to use this path to assign the predicted data packets to conduct pre-caching. To enable DPs or DCs to construct this path after receiving an interest packet, CRs and users would add their sequence number to an interest packet when they forward it, so the $\mathcal{P}_{cache}$ is recorded in the Selecters field.

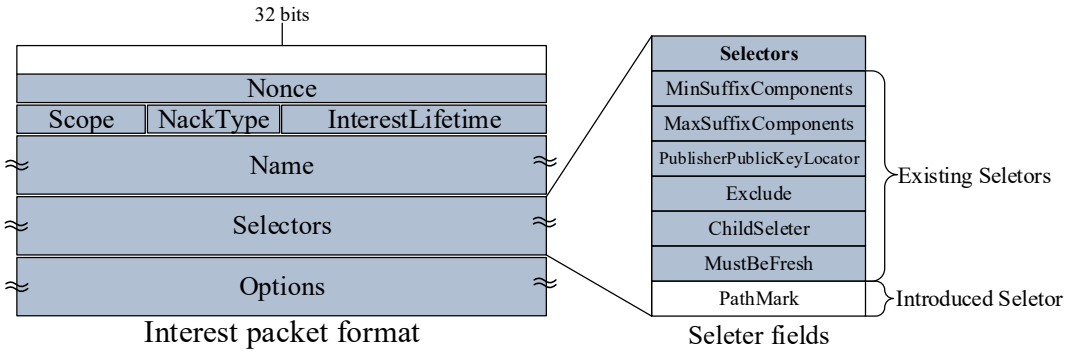

Figure 3. Interest packet format in the CJSR scheme.

(2) Data packet format. A data packet represents some arbitrary binary data held in the Content element together with its Name, some additional bits of information (MetaInfo), and a digital Signature of the other three elements. The Name is the first element since all NDN packet processing starts with the name. Signature is put at the end of the packet to ease the implementation because signature computation covers all the elements before it. To assign each data packet to its correct location in pre-caching, we add a PathMark field to data packets as it is shown in the white field of Figure 4. This field is used as the indication of a data packet's destination. Thus, P_m would be added to the PathMark field of a produced data packet; when a CR receive a data packet, it checks whether the P_m is same as its own sequence number and forwards this packet to next CR along the $\mathcal{P}_{cache}$ if the P_m does not matched.

(3) Extra Storage (ES). Figure 5b illustrates this data structure. To address the scarcity of storage space, we introduced the ES to use idle CRs. The structure and storage capacity of the ES and the CS is similar, each "extra chunks" is mapped to an "ename". When a CR on the $\mathcal{P}_{cache}$ asks an idle CR to cache a data packet on its behalf, relevant information is stored in the "ename" and "extra chunks" entries of the ES for future reference.

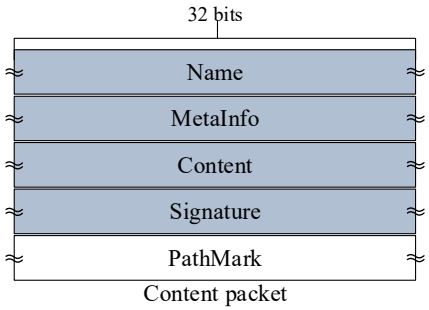

Figure 4. Data packet format in the CJSR scheme.

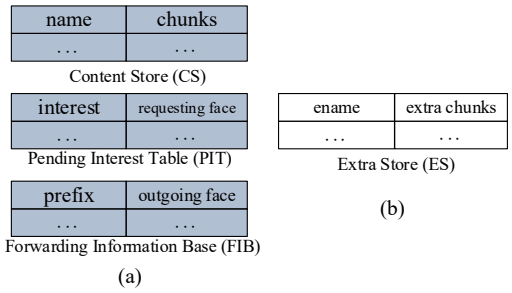

Figure 5. (**a**) NDN data structures; and (**b**) introduced data structure.

4.2. CJSR Scheme

The CJSR scheme is composed of Algorithms 1 and 2. The former gives detailed procedures of pre-caching, which roughly contains two steps. Step 1 describes how to process an interest packet and the assignment of data packets. Step 2 is the transmission process of a data packet. Algorithm 2 is the overall algorithm of the CJSR scheme, and gives the pseudo-code of the whole process of the cooperative pre-caching, which consists of uploading pre-caching and downloading pre-caching. In Section 4.3, we discuss a further attempt to address the scarcity of storage capacity.

The CJSR scheme could cache semi-real-time data in CRs intelligently by using the cooperative pre-caching. Moreover, with two shortcut routing methods, the length of routing path for users to obtain data is noticeably decreased. In the CJSR scheme, DPs not only collect and provide data, but also assign data towards users proactively. During the whole process, if cooperative pre-caching is conducted successfully, users could fetch their interested data from nearby CRs rather than DCs. That is to say, interest packets could be handled by CRs that have the requested data cached, thus the burden of processing interest packets would be decreased. In addition, it is also unnecessary for data packets to travel through the whole $\mathcal{P}_{cache}$ when they are requested by users. Therefore, the burden of forwarding data packets would also be declined, due to the decline of hop counts between CRs with requested data cached and users; latency could have a noticeable drop. Thus, Equations (1)–(3) could be improved, which provides an empirical guarantee for better QoS.

As a significant component of the CJSR scheme, the pre-caching algorithm is discussed initially. First, a user sends an interest packet I_m requesting for S_m (e.g., a real-time video relevant to breaking news). As I_m are being forwarded towards DCs or DPs (these two situations are discussed in the cooperative pre-caching of Algorithm 2), each CR adds its sequence number to the PathMark of

I_m. Then, the $\mathcal{P}_{cache}$ is constructed by the time this interest packet arrives at a DC or a DP. Because the same video streaming is successive, after requesting for S_m, this user is highly likely to request the subsequent data of the same streaming. In addition, due to the extensive public concern for breaking news, other users also have high probability to request the same data. Therefore, after the formation of $\mathcal{P}_{cache}$, pre-caching is triggered and the rest of data stored in the DC or the following data collected by the DP would be assigned to CRs along the $\mathcal{P}_{cache}$ providently. To assign unrequested data, the synthetic interest generator (discussed in NFD developer's guide) could be used to make entries in the PIT. In other words, from the CRs' point of view, the remaining data look like they have been requested. Thus, by the time those data packets arrive at vertices they are assigned to, they can be cached. Due to the limited storage size of each CR, each CR can only store a proportion of the data, thus the remaining data from $S_n = \{S_1, S_2, \ldots, S_m, \ldots, S_n\}$ are divided into a few arrays. The size of each array depends on the cache capacity of CRs (size of each array = C_{ξ_i} = d data packets). In the order from S_m to S_n, the remaining data are divided into (m − n)/d arrays. The first array which has the largest probability to be requested later is cached in the nearest CR (ξ_{r_i}) next to a user (r_i). Following that, each of the remaining arrays is cached into each CR along the $\mathcal{P}_{cache}$. Hence, each round of pre-caching can assign d$\times N_\xi$ data packets and the CJSR scheme is consisted of (m − n)/d $\times$ N_ξ round(s) of pre-caching. It is noticeable that only the interest packet requesting for the first data of each round can trigger pre-caching (I_m, $I_{m+d\times N_\xi}$, $I_{m+2\times d\times N_\xi}$ etc.), other interest packets would be processed by intermediary CRs rather than DPs or DCs if the pre-caching is conducted successfully. Nevertheless, an interest packet may not be satisfied by overlapped CRs (such situations are also considered in Scenario 2 and Scenario 4 of experiments); this interest has to be forwarded to the DP or the DC along the caching path and the data packet is sent back using LCE cache strategy. The specific algorithm of pre-caching is shown in Algorithm 1.

Algorithm 1 Pre-caching

Step 1: I_m requesting for S_m from $S_n = \{S_1, S_2, \ldots, S_m, \ldots, S_n\}$ is issued by a user
I_m arrives at a DP or a DC
Construct the $\mathcal{P}_{cache} = \{\xi_{i1}, \xi_{i2}, \ldots, \xi_{ij}\}$ from the PathMark element of I_m
Compute the number of vertices in the $\mathcal{P}_{cache}$: N_ξ
(n–m) packets are divided into (m–n)/d arrays, each of which contains d packets
For j = 1 to N_ξ do
  For k = m + j × d − d to m + j × d − 1
    Prouced a data packet S_k, add j to its PathMark fiekd indicating its desination
    Forward S_k to ξ_{ij}
  End For
End For
Step 2: *The transmission of a data packet* S_k
For k = m to n
  For j = N_ξ to 1
    S_k arrives at ξ_{ij}
    If S_k is found in the PIT of ξ_{ij}
      If the P_m of S_k is identical to ξ_{ij}'s sequence number
        Cache S_k
      Else
        Forward S_k to next vertex
      End if
    Else
      Drop S_k
    End If
  End For
End For

Next, the CJSR algorithm is given below. Using the pre-caching discussed in Algorithm 1, the cooperative pre-caching is achieved by applying pre-caching in uploading and downloading process in two possible situations: (1) The requested data are stored in a DC. Then, the DC could perform downloading pre-caching between the user and itself. During this process, interest packets are forwarded using the prefix of the DC. Following that, the chunk is produced using DC's signature. (2) The requested data have not been uploaded yet. In this case, the DC would respond the user's request with the location of the DP that is collecting the requested data. Once the user gets the address of that DP, the request can be forwarded to the DP directly, and then the DP could perform uploading pre-caching. The user would use the DP's prefix to request, while the produced data packets are signed by the DP. In other words, each chunk only has one name in these two situations, hence cooperative pre-caching is applicable for various situations. Another main idea of the CJSR scheme is the formation of shortcut. After acquiring the location of the DP, the user directly request data from the DP, since now the first kind of shortcut is constructed. In addition, the uploading process and pre-caching happens at the same time, and the uploading path of data packets would share some segments of $\mathcal{P}_{cache}$. Then, data packets would find a CR that could yield shorter overall routing path to form the second kind of shortcut. Firstly, prefixes from Data Centers are used if users request data from DCs. After the formation of the first shortcut, the prefix of the Data Producer is used and interest packets are sent to the DP. By combining the cooperative pre-caching and two shortcut routing methods, the burden of all devices (especially DCs and DPs) could be decreased significantly, cache hits would ascend and latency would drop. Algorithm 2 depicts the concrete process of the CJSR scheme.

Algorithm 2 Caching Joint Shortcut Routing

I_m requesting for S_m from $S_n = \{S_1, S_2, \ldots, S_m, \ldots, S_n\}$ is issued by a user
I_m arrives at a vertex
If the pre-cached data in vertex ξ_{ij} meets the demand after performing lookup on its Content Store
  Forward S_m to user
Else
  Forward I_k to next vertes
  If I_m arrives at a DC
    If the requested data have not been uploaded
      The DC responds the location of DP to the user
      The user forwards i_m towards the DP
      The DP performs algorithm 1 between the DP and the user (uploading pre-caching)
    Else
      The DC performs algorithm 1 between the DC and the user (downloading pre-caching)
  End If
End If

Next, the whole process of the CJSR scheme is illustrated using a concrete example along with a series of detailed figures. We make some modifications to the scenario we used in Figure 1 and use it as our example. It is assumed that there are one user (another user will join the network later), one Data Producer (DP), one Data Center (DC) and n CRs. When breaking news happens, a user issues an interest for a short video about this news and hopes to obtain the data with high QoS, a DP is collecting and sensing data relevant to the news, and a DC is waiting for the data to be uploaded. Moreover, we assume that the requested streaming contains 20 data packets, $N_\xi = 5$, and the storage size of each CR is 4 data packets. At the beginning, the user is interested in S_1 and requests it. The forwarding of I_1 automatically find the shortest route to a DC. To focus on routing paths, we omit the links among CRs in the following graphs. Detailed procedure and ideas are:

(1) Initial process of requesting data. In Figure 6, a DP is collecting data about that news, then uploads data to a DC according to its public prefix (Routing Path ① in Figure 6). Then, the user requests S_1. By that time, the user does not possess any information about where the data are produced,

so the interest packet is sent to the DC (Routing Path ② in Figure 6). Because the requested data S_1 have not been uploaded, the DC responds with the prefix of the DP that is collecting the requested data to the user (Routing Path ③ in Figure 6). As a result, uploading pre-caching of the cooperative pre-caching mechanism is going to be performed in this example.

(2) Re-direction of data request. After receiving the prefix information about the DP, the user starts to request data from the DP directly (Routing Path ② in Figure 7).

(3) The formation of the first shortcut routing path. As it is shown in Figure 8, a transmission path between the DP and the user is built thus the user does not have to obtain data from the DC. In this case, the shortcut is also the $\mathcal{P}_{cache}$. The number of CRs (N_ξ) on the $\mathcal{P}_{cache}$ is five. Because of the scarcity of storage size, S_1 to S_{20} is divided into five arrays (as it is demonstrated in the left part of Figure 8), then the first round of pre-caching is conducted and five arrays are cached in five CRs along the $\mathcal{P}_{cache}$.

(4) Pre-caching on the uploading path of data. Routing Path ② in Figures 9–13 demonstrates the whole procedure of uploading pre-caching. Because the subsequent part of the same streaming is likely to be request later, especially the first array. Besides S_1, S_2 to S_4 are more likely to be requested later than other arrays. Hence, the first array is assigned to Vertex E. S_1 is sent to user as soon as S_1 is received so the first request of user is satisfied by that time and the following pre-caching are carried out. S_5 to S_8 are assigned to Vertex D, of which the hop counts to user is two. Similarly, S_9 to S_{12} are cached in Vertex C, of which the hop counts to user is three; S_{13} to S_{16} are cached in Vertex B; and S_{17} to S_{20} are cached in Vertex A. The storage of each CR is shown in Figure 14.

(5) The formation of the second kind of shortcut. Initially, the data are uploaded as shown in Routing Path ① of Figure 8. After the formation of the first kind of shortcut, data packets pass through the $\mathcal{P}_{cache}$ while they are uploaded as it is depicted in Routing Path ① of Figure 9, so that they are cached into CRs while they are uploading. During the conducting of pre-caching, the back proportion of the remaining data are assigned to the CRs that are far away from user. Data packets would adjust its shortcut to find a CR that could yield shorter overall routing path, then upload data through that chosen CR to the DC. Firstly, the chosen CR is Vertex D (Figure 10). Then, the uploading path of data packets follows Routing Path ① in Figure 11. Following that, the uploading path is shown as Routing Path ① in Figure 12. Finally, data packets upload through Vertex A (Routing Path ① in Figure 13).

(6) Request of cached data (the user's requests). Rather than forward each interest packet to DP in a time-consuming way, interest packets could be handled by CRs that have the requested data cached.

(7) The participation of other users. There may be two possible situations after User 2 joins the network: (i) User 2 requests for different data. Such a scenario is demonstrated in Figures 15–17. Firstly, User 2 requests S_k from the DC (Routing Path ② in Figure 15). It is reasonable that User 2 may request the other relevant data of that breaking news (also have not been uploaded yet) from another DP. Since the requested data are different, two users would conduct the CJSR scheme separately. Because User 1 and User 2 are both located on the network edge, it is assumed that they are connected to Vertex E and Vertex F respectively. After requesting from the DC, User 2 gains the prefix information of DP 2 and sends request to it (a new caching path is established as shown in Caching Path ② in Figure 16). After constructing this pre-caching path, the two users' pre-caching path overlap at Vertex C as demonstrated in Figure 17. When User 2 starts to request data, a new round of pre-caching is triggered, and all the data cached in the overlapped CR (Vertex C) would be replaced. This scenario is evaluated in Scenario 4 in Section 5. (ii) User 2 requests for data that has been uploaded. Figures 18 and 19 depict that User 2's request are satisfied by the DC, although the requested data may still be held by a CR. Since the first interest packet of User 2 is forwarded to the DC, the pre-caching path is built between the DC and the User 2 and the downloading pre-caching is triggered. Following that, the DC would perform downloading pre-caching (similar procedures demonstrated in Figures 9–13 are omitted).

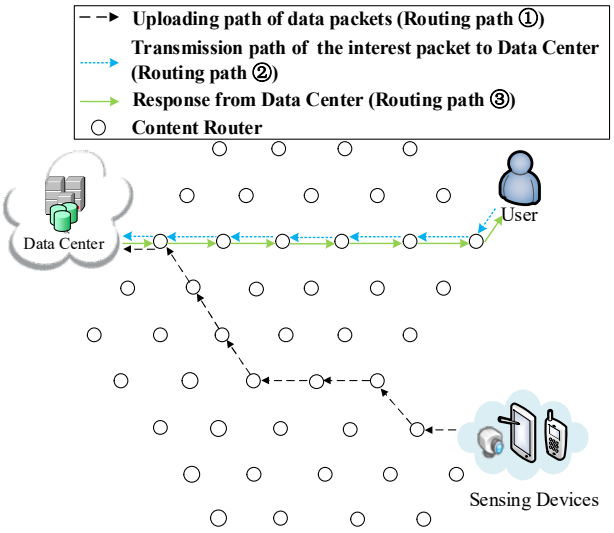

Figure 6. The exchange of information between the user and the DC.

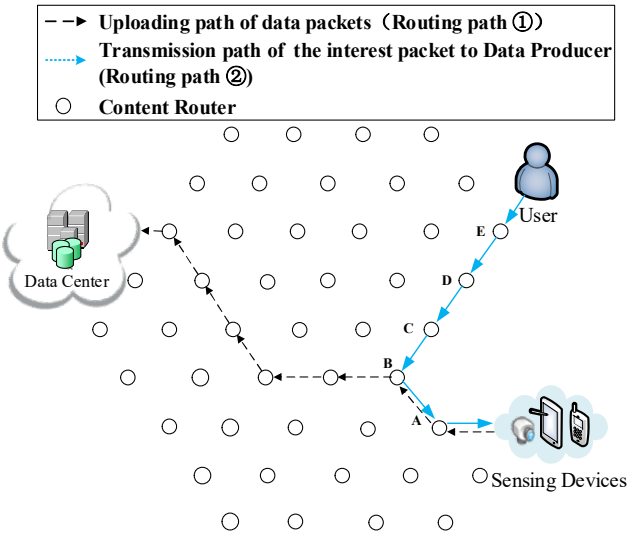

Figure 7. User start to request data from the DP.

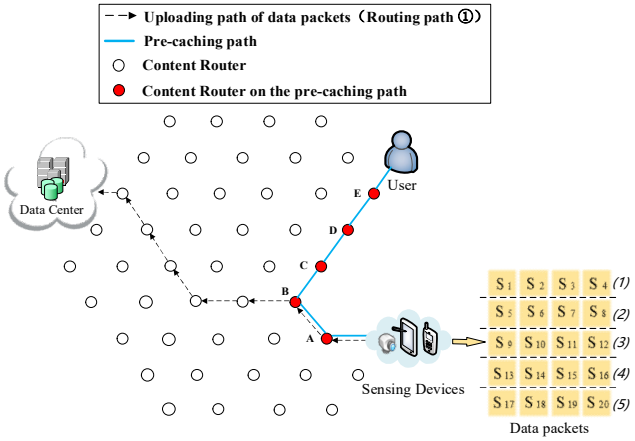

Figure 8. The formation of $\mathcal{P}_{cache}$.

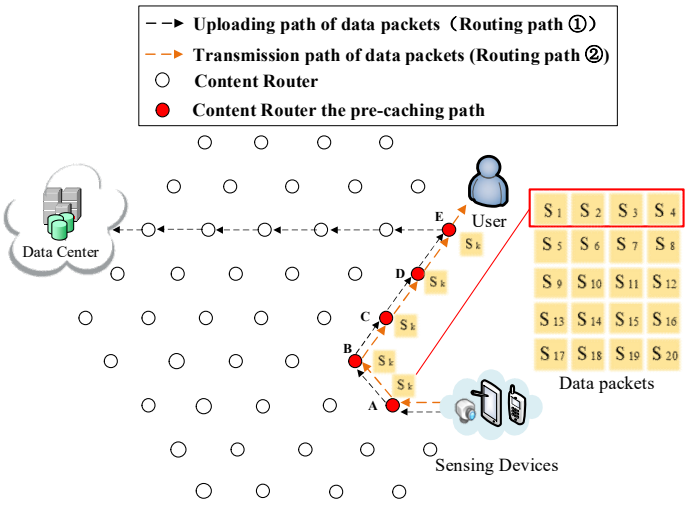

Figure 9. Send the first array S_k (k = 1, 2, 3, 4) to Vertex E.

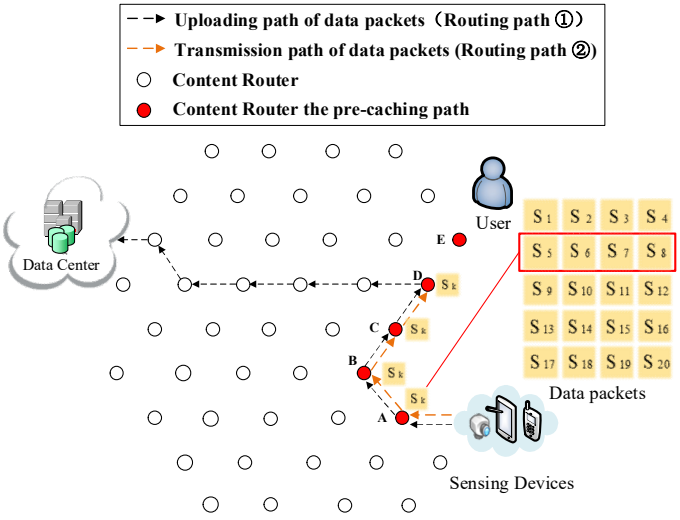

Figure 10. Send the second array S_k (k = 5, 6, 7, 8) to Vertex D.

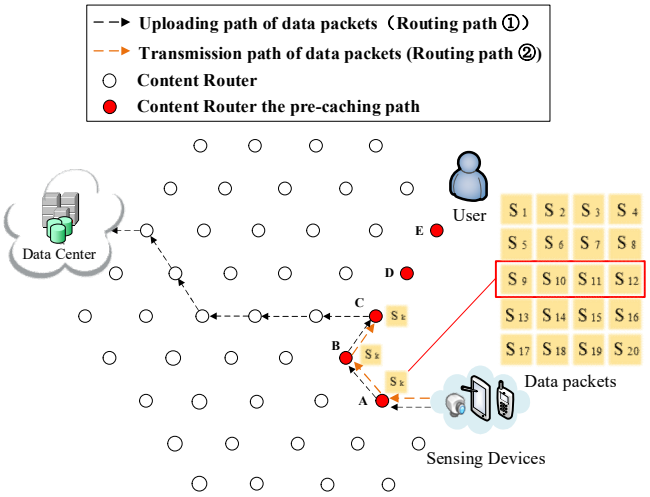

Figure 11. Send the third array S_k (k = 9, 10, 11, 12) to Vertex C.

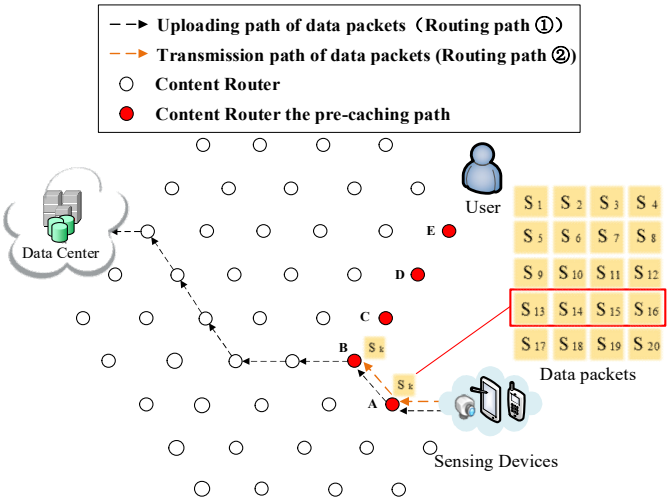

Figure 12. Send the fourth array S_k (k = 13, 14, 15, 16) to Vertex B.

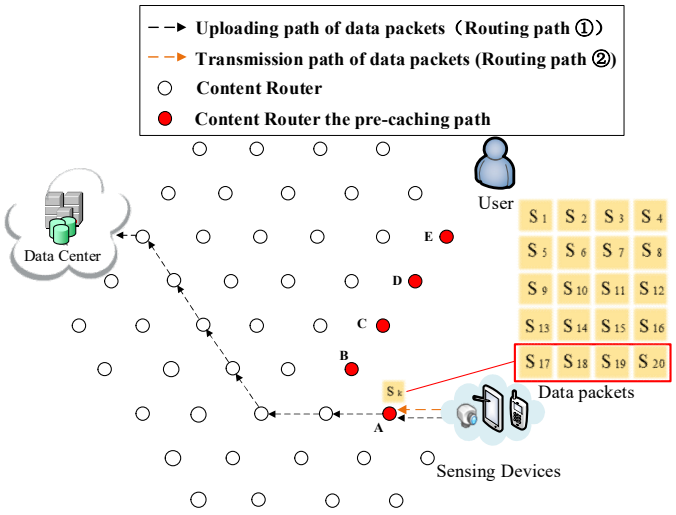

Figure 13. Send the fifth array S_k (k = 17, 18, 19, 20) to Vertex A.

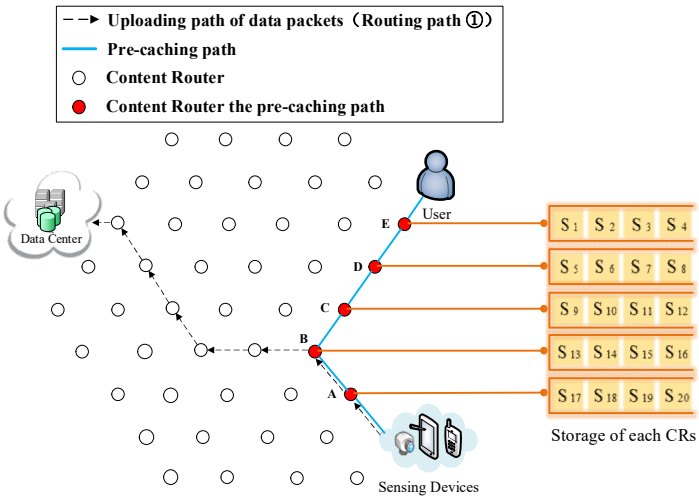

Figure 14. The cache storage of each CR along the $\mathcal{P}_{cache}$.

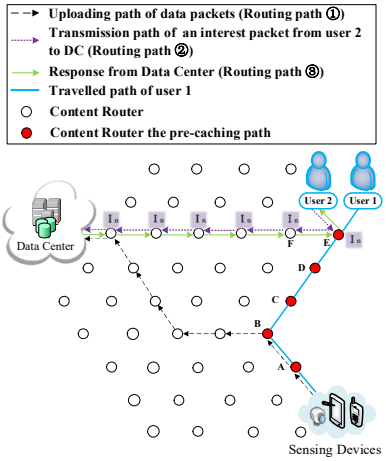

Figure 15. User 2 joins the network demanding different data.

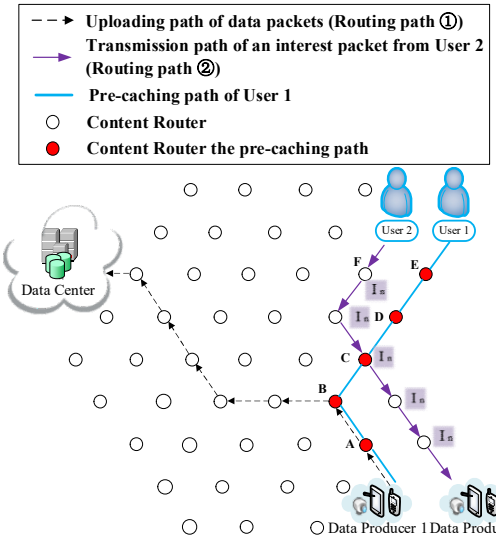

Figure 16. User 2 starts to request from the DP.

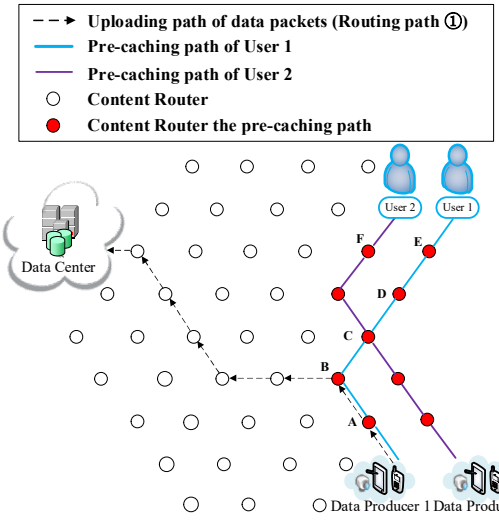

Figure 17. Two users build up their own caching path.

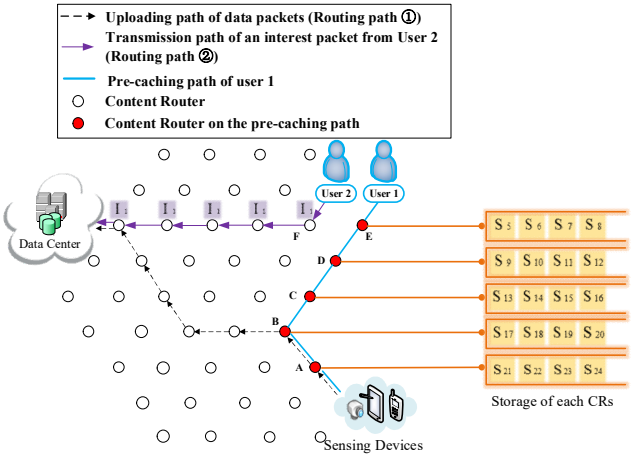

Figure 18. User 2 joins the network and requests the same data that User 1 had requested.

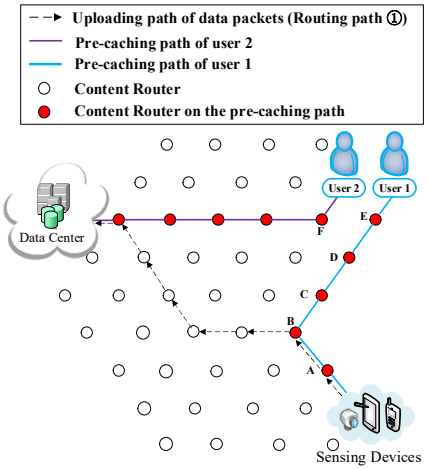

Figure 19. The formation of User 2′s $\mathcal{P}_{cache}$.

4.3. Idle CRs Involved Pre-Caching (ICIP)

In Algorithm 1, data packets are assigned to CRs along the $\mathcal{P}_{cache}$, which could be downloading routing paths from DCs to users or uploading routing paths from DPs to users. However, during the whole process of the CJSR scheme, only those CRs on those fixed paths are actively involved in pre-caching. To exploit the storage resource from underused idle CRs (we define a CR that is not on the $\mathcal{P}_{cache}$ as an idle CR) and help CRs on the caching path to cache the data that they cannot accommodate, Algorithm 3 is proposed. In this algorithm, Step 1 is about processing an interest packet. What is different from Algorithm 1 is that the frequency of an interest packet is recorded and used as the criteria in judging the importance of its corresponding data. Step 2 describes the procedures of processing a data packet at each CR.

The main idea in Algorithm 3 is to address the scarce caching storage in an indirect method. In the first place, a standard should be made to judge the priority of a data packet. If the number of interest packets requesting for a data packet is high, these data are more popular, and may be related to a hot issue or breaking news. Therefore, each CR records a frequency of an interest packet ($\mathcal{A}^{m}_{\xi_{i,r_j}}$) and uses it as the criterion. For a newly arrived data packet, comparisons of the Interest frequency are being made before asking help from an idle CR. If the Interest frequency of the incoming data are higher than that of one or more stored data packets, the least frequently used data packet is replaced by the new one. If all the Interest frequency of cached data packets are higher than that of the incoming data packet, which means that all of the stored data have high probability to be requested sooner, this CR will ask

an adjacent idle CR to store that unaccommodated data. Here is the detailed process: after a period of pre-caching, the storage of each CR on the $\mathcal{P}_{cache}$ is full. When a new data packet arrives at ξ_{ij}, the $\mathcal{A}^{m}_{\xi_i,r_j}$ of each cached data packet is compared and the least frequent used data are replaced. If the $\mathcal{A}^{m}_{\xi_i,r_j}$ of cached data packets are all high, the newly arrived data packet is forwarded to an adjacent idle CR ξ_{idle}, and store ename and extra chunks into the ES. When user requests that data, CRs would check the ES of their adjacent CRs if the data are not found after performing traditional lookup. Using the PathMark field of data packets, if a CR cannot accommodate a data packet, the original sequence number indicating its destination is modified into a special mark, then this data packet is broadcasted to neighbors. Because no CR along the caching path matches the PathMark field of the data packet, this data packet is immediately dropped and only an idle CR could cache it. The detailed process is shown in the following Algorithm 3.

Algorithm 3 Idle CRs Involved Provident Caching (ICIP)

I_{m} requesting for S_m from S_n= $\{S_1, S_2, \ldots, S_m, \ldots, S_n\}$ is synthesized by user r_j
Step 1: *I_m arrives at ξ_{ij}*
$\mathcal{A}^{m}_{\xi_{ij},r_j}$++
If S_m is found after performing lookup in the Content Store
  Forward S_m to r_j
Else
  Check ES
  If found entry in ES
    Forward S_m to r_j
  Else forward I_m to next vertex
    If I_{m} arrives at a DC or a DP
      Perform algorithm 1
    End If
  End If
End If
Step 2: *A data packet S_k arrives at a CR ξ_{ij}*
If I_k is found in the PIT of ξ_{ij}
  If the P_m of S_k is identical to ξ_{ij}'s sequence number
    If the left cache storage of ξ_{ij} is enough
      Cache the S_k
    Else
      For each cached data packet S_i in ξ_{ij}
        If $\mathcal{A}^{k}_{\xi_{ij},r_j} > \mathcal{A}^{i}_{\xi_{ij},r_j}$
          Replace the least frequently used data packet S_i with S_k
        Else
          Forward the S_k to ξ_{idle}
          Add S_k to Extra Store
        End If
      End For
    End If
  End If
Else
  Drop S_k
End If

The procedure of Algorithm 3 is also illustrated using an example, in which the assumptions and parameters are same as they are in Section 4.1.

(1) Pre-caching process. After performing uploading pre-caching for a period, due to the limitation of caching storage (four data packets), storage capacity of each CR has been filled, as Figure 20 demonstrates.

(2) Replacement of a data packet. At some time while the DP is conducting pre-caching, the data packet S_{21} is assigned to Vertex E (Routing Path ② in Figure 21). Figure 22 demonstrates that, when S_{21} arrives at Vertex E, the $\mathcal{A}^1_{E,r_j}$, $\mathcal{A}^2_{E,r_j}$, $\mathcal{A}^3_{E,r_j}$ and $\mathcal{A}^4_{E,r_j}$ are compared with $\mathcal{A}^{21}_{E,r_j}$, respectively. As illustrated in Figure 23, the result of comparisons is that $\mathcal{A}^1_{E,r_j} < \mathcal{A}^{21}_{E,r_j}$, then S_1 is substituted by S_{21}.

(3) Using storage capacity of an adjacent idle CR. If all the Interest frequency of cached data packets in Vertex E ($\mathcal{A}^1_{E,r_j}$, $\mathcal{A}^2_{E,r_j}$, $\mathcal{A}^3_{E,r_j}$ and $\mathcal{A}^4_{E,r_j}$) are bigger than $\mathcal{A}^{21}_{E,r_j}$, S_{21} is forwarded to Vertex E (Routing Path ② in Figure 24). When S_{21} is requested, the Vertex F sends S_{21} to the user according to the Extra Storage (Routing Path ② in Figure 25).

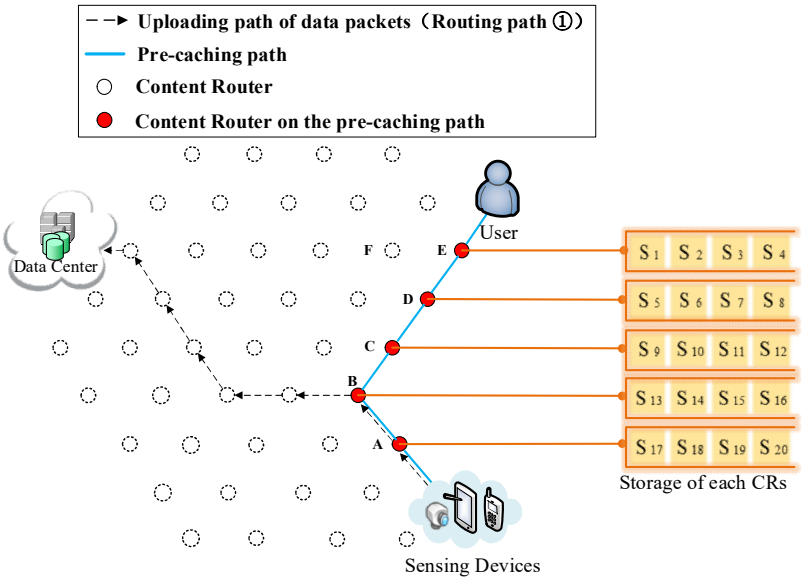

Figure 20. Cache capacity of each CR is full.

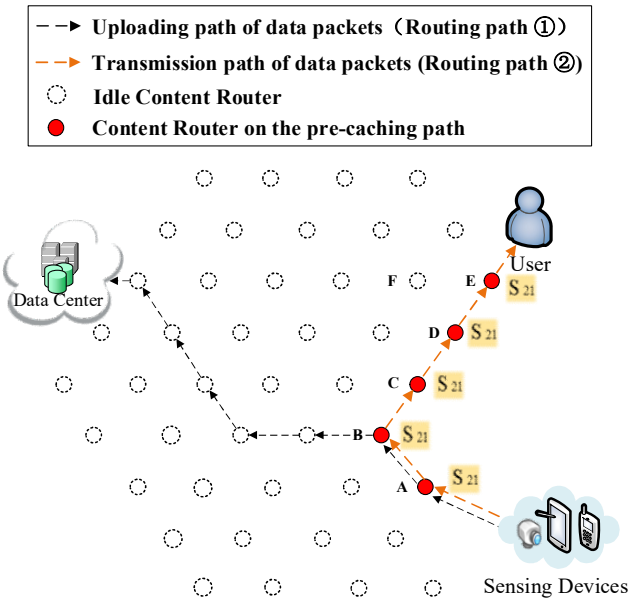

Figure 21. S_{21} is forwarded to Vertex E.

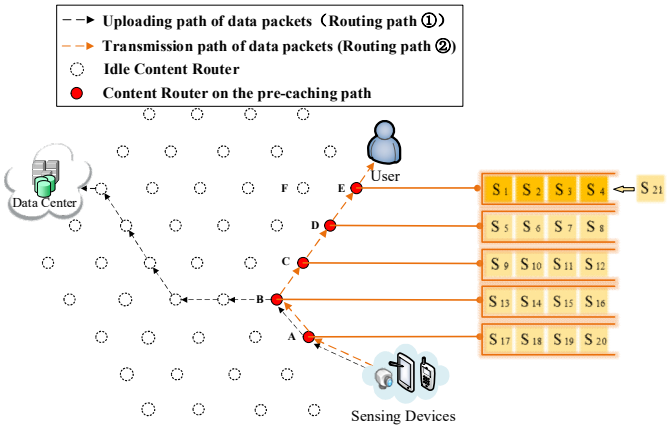

Figure 22. Comparisons of the Interest frequency.

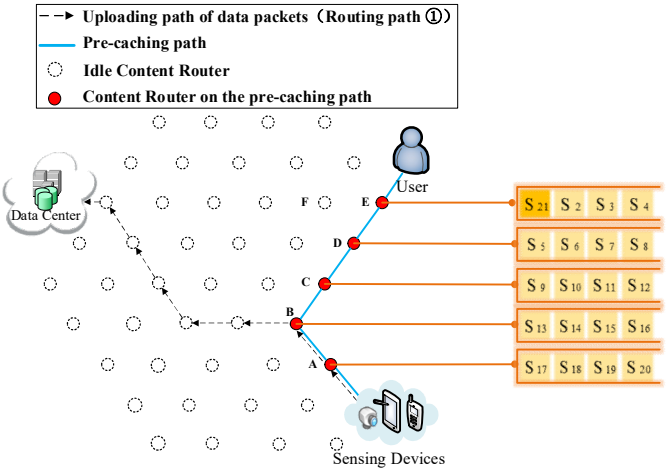

Figure 23. Replacement of S_1.

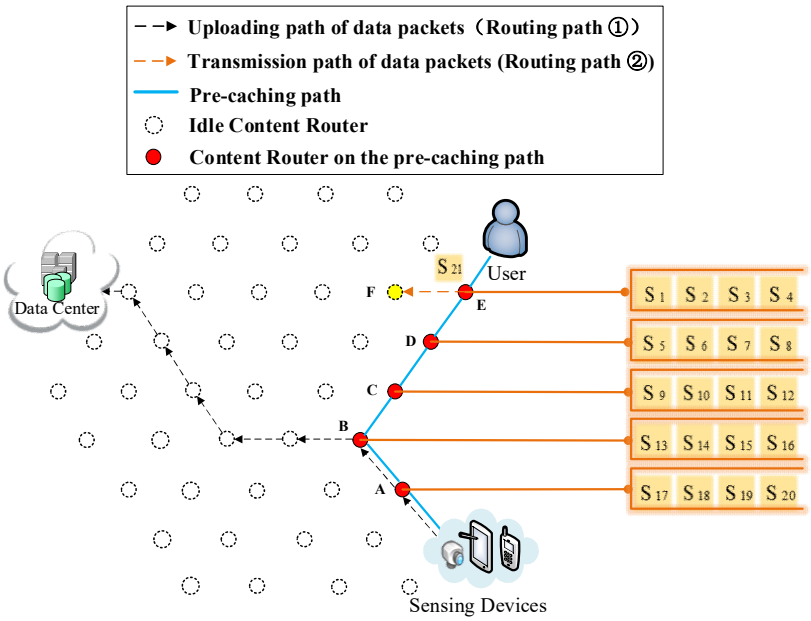

Figure 24. Forward S_{21} to Vertex F.

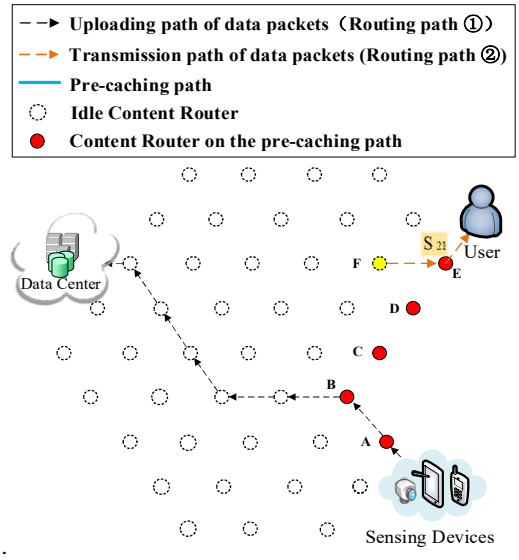

Figure 25. Forward S_{21} in respond to the user.

5. The Experimental Results and Analysis

In this section, extensive experiments were conducted and the performance of the CJSR scheme is evaluated under three different metrics. We made comparisons among two proposed schemes and five existing schemes: (1) Traditional NDN; (2) No Cache; (3) Probability Cache; (4) LCD; and (5) MCD. In the traditional NDN scheme, data packets are cached into each CR they have been forwarded. In contrast, CRs in the No Cache scheme can only forward packets. As for the probability Cache scheme, a CR will cache a data packet based on a predefined probability; the probability parameter is set to 0.5 in our experiments. The LCD and the MCD scheme share a similar idea which is to cache the data packet in the next vertex of the vertex where a cache hit occurs. However, in the latter scheme, the hit data cached in the CR where the cache hit takes place is deleted. Initially, the total hop counts are compared and analyzed in Section 5.1, which denotes the number of total hop counts of all the requested data packets. This criterion reflects the latency and is of critical importance to meet real-time requirement of many applications. Next, the number of processed packets on each CR along the caching path is evaluated in Section 5.2. The processed interest packets and the processed data packets denote the number of forwarded interest packets and incoming data packets respectively. Finally, comparisons of cache hits are made in Section 5.3.

After analyzing the experiment results, we have concluded that for the optimization targets Equations (1)–(3), the proposed mechanism guaranteed high performance in terms of the amount of processed packets, cache hits and total hop counts. High performance of these metrics above would contribute to a high QoS for users.

To test the proposed CJSR scheme under various scenarios, a topology deployed with 88 Content Routers (Figure 26) is produced for the experiments. The dotted lines denote that Users 2–12 are connected to different CRs in various scenarios. As shown in Figure 27, another larger topology (113 Content Routers) is generated for Scenario 6. Overall, to evaluate the uploading pre-caching of the CJSR scheme, all of the scenarios assume the following: (i) When users request data, their interested data have not been uploaded yet, so the first kind of shortcut is built between the DP and users; to focus on the routing between DPs and users, DCs are omitted in the topology. (ii) All the CRs are available and willing to cooperate. (iii) All users request successively and continuously.

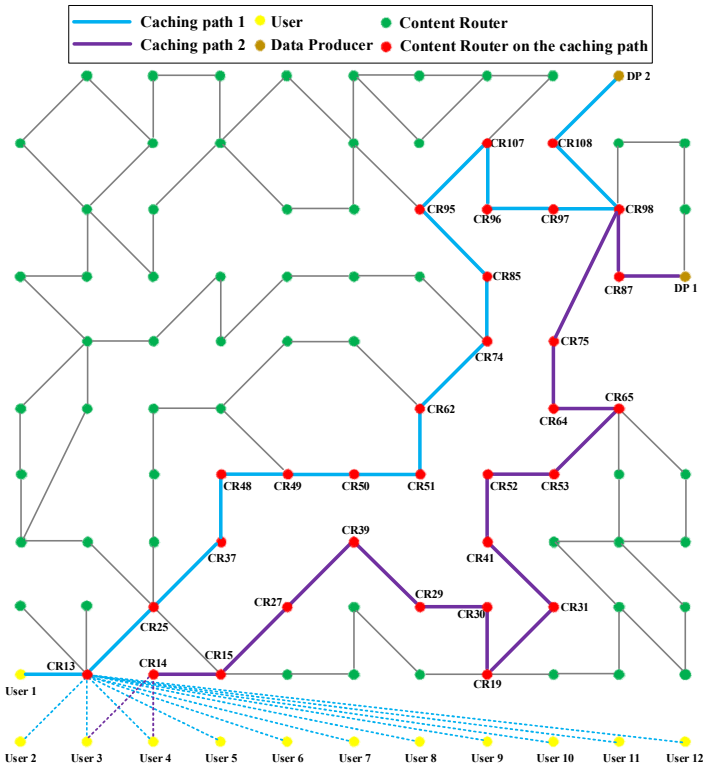

Figure 26. Network topology for Scenarios 1–5.

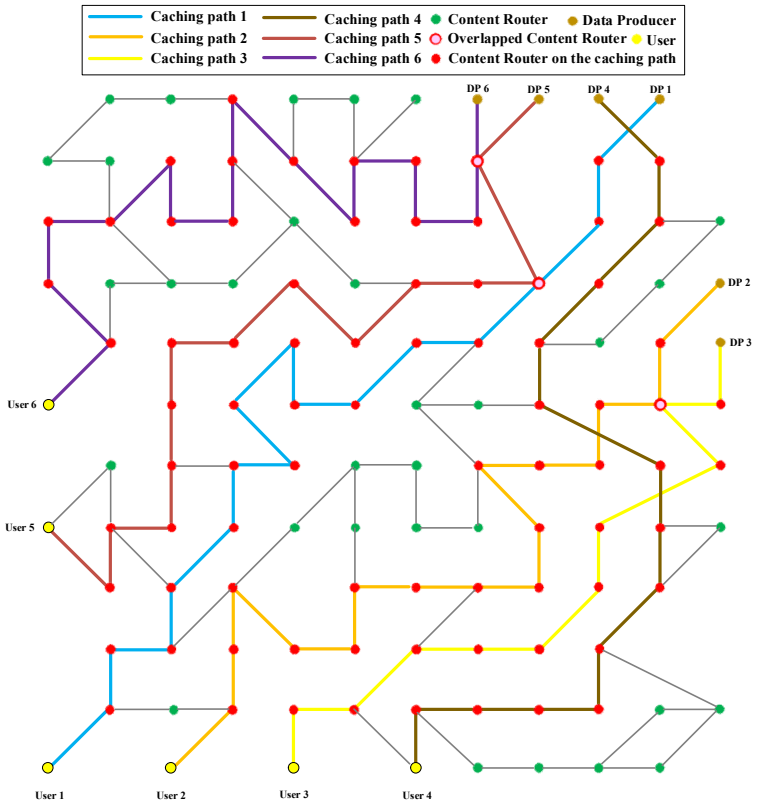

Figure 27. Network topology for Scenario 6.

Next, we discuss the design and configurations of the experiments. To test the network performance under various situations, six scenarios are designed. The detailed parameters and configurations can be found in Tables 2 and 3. Overall, the cache capacity is set to 250 so that each CR can accommodate 250 data packets, and the LRU replacement policy is used to replace old content in cache storage. In addition, each chunk has a Time-To-Live (TTL) to limit its lifespan if necessary. However, the configuration of TTL is highly application-dependent and the detailed evaluation on it is beyond the scope of this work. Hence, the TTL used in the experiments is set to be larger than 5 s (which is the time from a chunk is cached to it is replaced, so that each chunk remains stored in cache until it is replaced). The requested data contain 30,000 data packets, the interest frequency is 50 interests per second and simulation time is 10 min.

Table 2. Parameters.

Parameter	Values
Routing strategy	Automatic shortest routes
Interest frequency	50 interest packets per second
Cache replacement policy	Least recently used (LRU)
Forwarding strategy	Best-route
Simulation time	10 min
Connection	Point to point
Number of CRs (Scenarios 1–5)	88
Number of CRs (Scenario 6)	113
probability parameter	0.5
cache capacity	250

Table 3. Parameters of five scenarios.

Scenario	Number of DPs	Number of Users	Simulation Time	Total Number of Requested Data Packets
Scenario 1	1	1	User 1: 10 min	User 1: 30,000 packets
Scenario 2	1	2	User 1: 10 min User 2: 599 s	User 1: 30,000 packets User 2: 29,950 packets
Scenario 3	1	12	User 1: 10 min Users 2–12: 72 s	User 1: 30,000 packets Users 2–12: 3600 packets
Scenario 4	2	2	User 1: 10 min User 2: 10 min	User 1: 30,000 packets User 2: 30,000 packets
Scenario 5	2	4	Users 1 and 3: 10 min Users 2 and 4: 50 s	Users 1 and 3: 30,000 packets Users 2 and 4: 2500 packets
Scenario 6	6	6	User 1–6: 10 min	Users 1–6: 30,000 packets

Scenario 1: In this scenario, one user is deployed. Experiment results reveal that most of interest packets are processed by intermediary CRs. However, the amount of processed data only had a slight decline because the assignments of data would produce some traffic. However, when more than one user request the same data, as in Scenario 3, the traffic of data packets would decline because data assignments only have to occur once, then only the transmission of data packets from CR holding the requested packets to users would produce some traffic.

Scenario 2: User 1 and User 2 request same data from DP 1. User 2 starts to request one second later than User 1. Requesting later, User 2 can directly benefit from the cached data. In the beginning of the second round pre-caching, while User 2 requests S_{3949} to S_{3999}, those data packets have just been replaced by the assigned data from User 1 (S_{4000} to S_{4049}) prior to User 2's request. Therefore, User 2 has to send I_{3949} to I_{3999} to the DP. Similarly, User 2's requests have to be forwarded to the DP in the remaining rounds (the whole process consists of eight rounds of pre-caching). In contrast, however, the request of replaced data from User 2 (S_{3949} to S_{3999}, S_{7949} to S_{7999} etc.) from User 2 can be satisfied in the ICIP scheme due to the involvement of idle CRs. Because each user requests consecutive data, the interest frequency of data packets are same. As a result, when the assigned data from User 1 arrives at CR13, no replacement occurs and CR13 forwards newly arriving data packets (e.g., S_{4000} to S_{4049}) to

an idle CR. When User 2 requests S_{3949} to S_{3999}, his interest can still be satisfied. Nevertheless, if User 2 starts to request five seconds later, which is the time that cache replacements occur, User 2 cannot benefit from cached data at all.

Scenario 3: Twelve users (Users 1–12) request same data from DP1. After the first user discovers data relevant to breaking news, same data may be requested by many users at the same time. This scenario depicts such a situation and Users 2–12 are started at the same time one second latter than User 1. In this scenario Caching Path 2 is choose to evaluate performance.

Scenario 4: Two users request different data from two DPs. In this scenario, users conduct their own caching schemes separately. Nevertheless, when the CJSR and the ICIP schemes are applied, the caching paths of two users may overlap because the form of caching path is the travelled path of the interest that triggers the pre-caching. As shown in Figure 26, the two caching paths overlapped at CR 98; as a result, the overlapped CR has to undergo more traffic burden. The overlap would also have negative influence for the CJSR scheme. In each round of pre-caching, the data packets assigned to the CR 98 are replaced by each other, so that interest packets cannot be satisfied by CR 98; then they are forwarded to the DP; and LCE cache policy is used for the remaining data which could have been satisfied by the CR 98 if there were only one user. Following that, all the cached data of CRs along the caching path was replaced due to those requests. For instance, in the first round, S_0 to S_{3499} from User 1 and User 2 are satisfied by assigned data. Next, I_{3499} to I_{3749} have to be sent to the DP, and all the cached data packets of CRs along the caching path are replaced by S_{3499} to S_{3749} because they are sent back using LCE policy. Then, the remaining requests of first round (I_{3749} to I_{3999}) have to be sent to the DP. Therefore the overlapped CR has a negative impact on the CJSR scheme, however, the ICIP scheme is not influenced. Conversely, during the data assignments of the ICIP scheme, the conflicts of data replacement are solved by the involvement of an idle CR. Thus, both users enjoy assigned data throughout each entire round. Caching Path 2 is used to demonstrate results.

Scenario 5: User 1 and User 2 request same data from DP 1 while User 3 and User 4 are requesting same data from DP 2. User 1 as well as User 3 start to request at the same time, then User 2 and User 4 are started one second later. Similar to Scenario 4, the rest of interest packets are forwarded to the DP from the moment their interest packet arrives at CR 98.

Scenario 6: To comprehensively test the proposed algorithms, another topology is used in this scenario as shown in Figure 27. In addition, there are six DPs producing various data for six users. While the CJSR scheme is conducted, six caching paths are built separately. Similar to Scenario 4, sometimes caching paths overlap and the CJSR scheme would not perform as good as it does in other scenarios due to the conflicts of data replacement. However, those conflicts have no impact on the ICIP scheme. Finally, Caching Paths 3, 4 and 6 are chosen to depict the experimental result because they represent different length and two of them contain CRs that shared by multiple caching paths.

5.1. Delay Analysis

As shown in the Figures 28–35, the given bar charts illustrate that the CJSR scheme reacts to user's request swiftly and the total hop counts of User 1 has reduced 51.6% compared with that of other schemes. According to Figures 29, 30 and 32, when there are more than one users, User 2′s total counts of the traditional NDN outperformed proposed schemes because User 2′s requests are satisfied by ξ_{r_1}(CR13) which has one hop count to the users. With lower total hop counts, high QoS could be guaranteed and the optimization objective of Equation (2) is achieved.

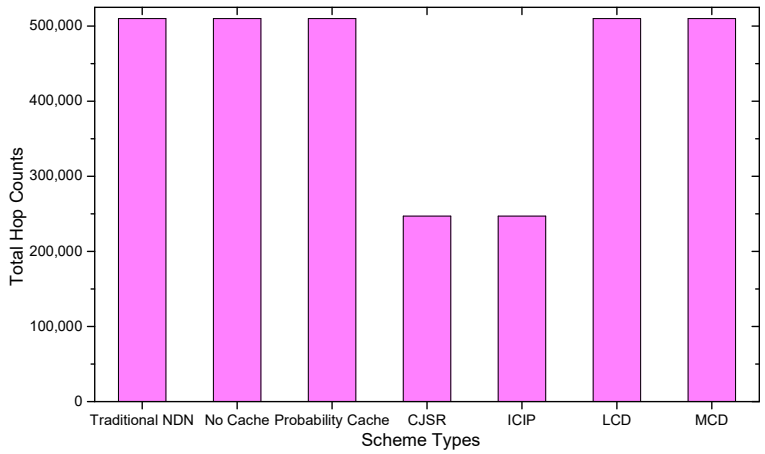

Figure 28. The total hop counts of all the requested data packets in Scenario 1.

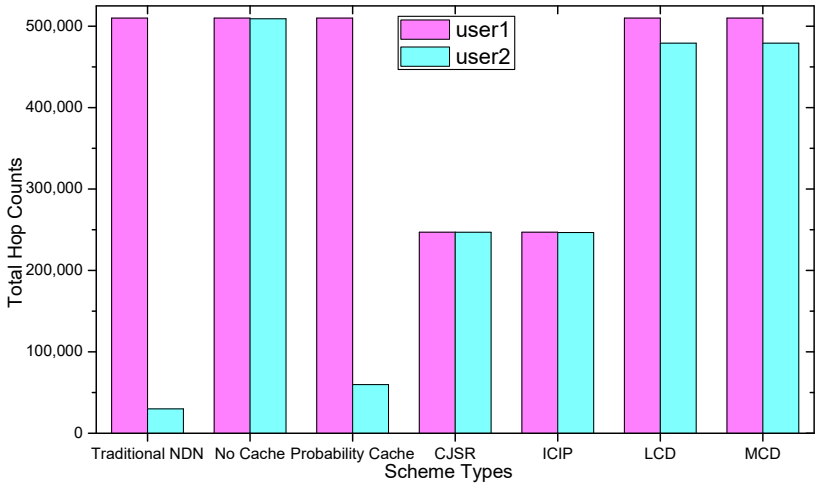

Figure 29. The total hop counts of all the requested data packets in Scenario 2.

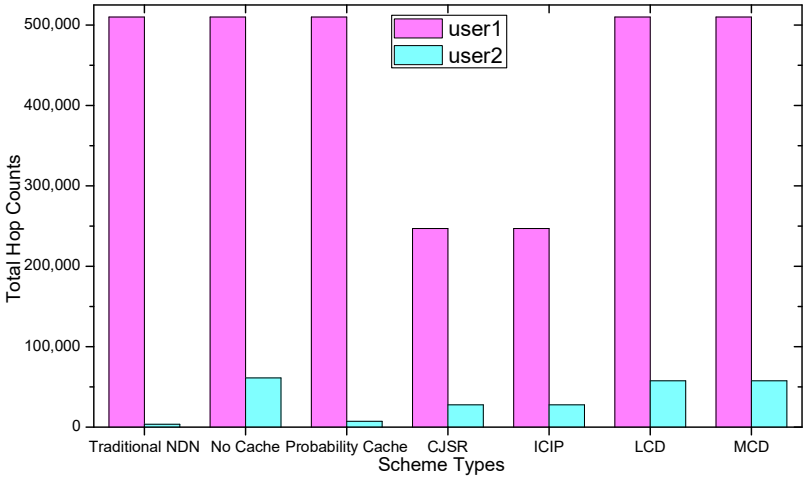

Figure 30. The total hop counts of all the requested data packets in Scenario 3.

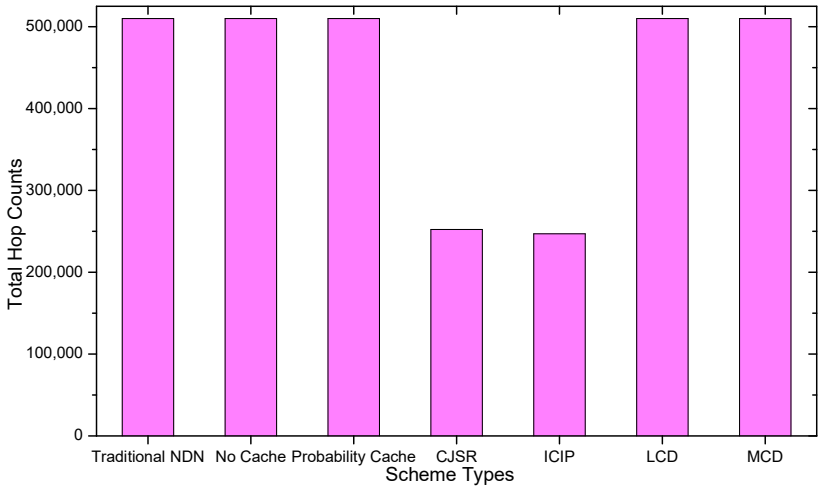

Figure 31. The total hop counts of all the requested data packets in Scenario 4 for both User 1 and User 2.

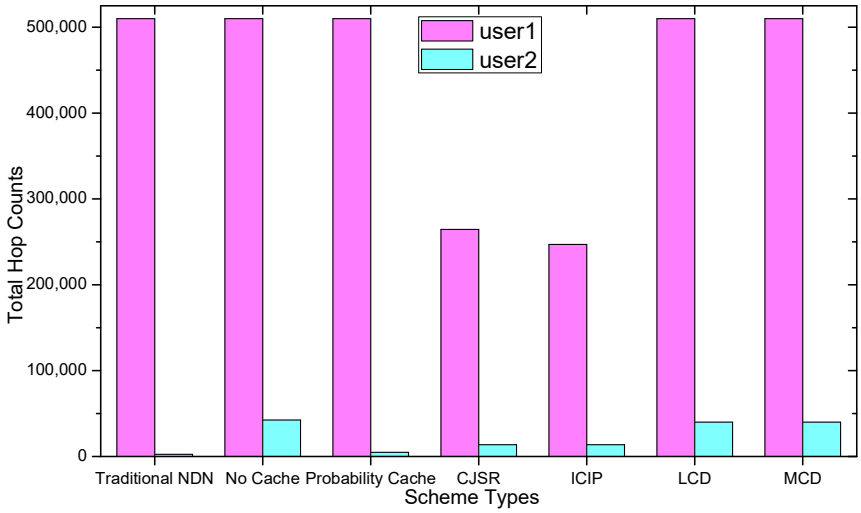

Figure 32. The total hop counts of all the requested data packets in Scenario 5.

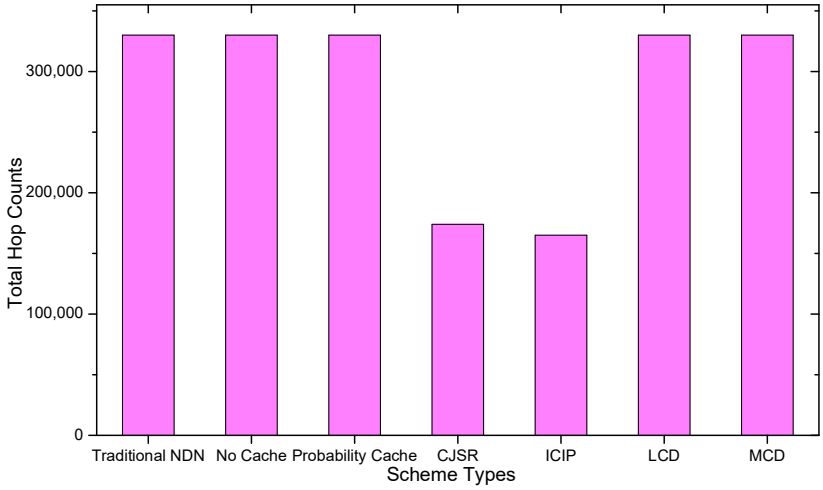

Figure 33. The total hop counts of all the requested data packets in Scenario 6 for User 3.

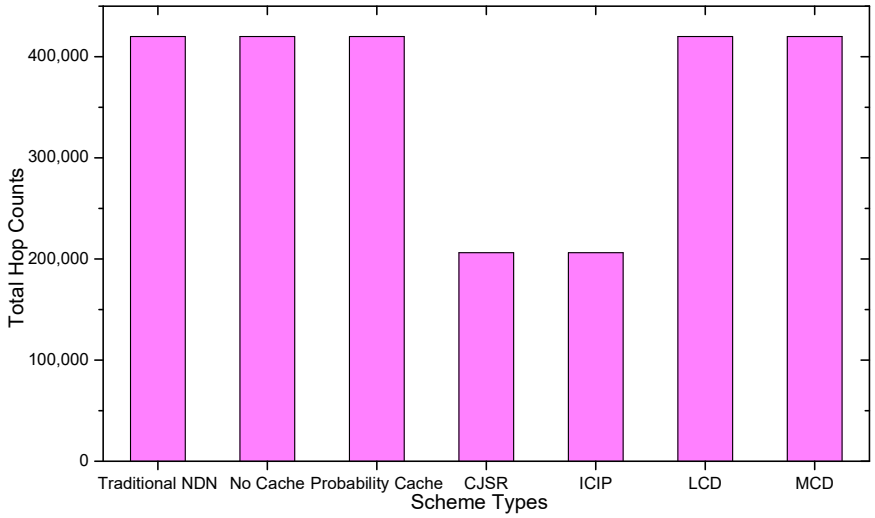

Figure 34. The total hop counts of all the requested data packets in Scenario 6 for User 4.

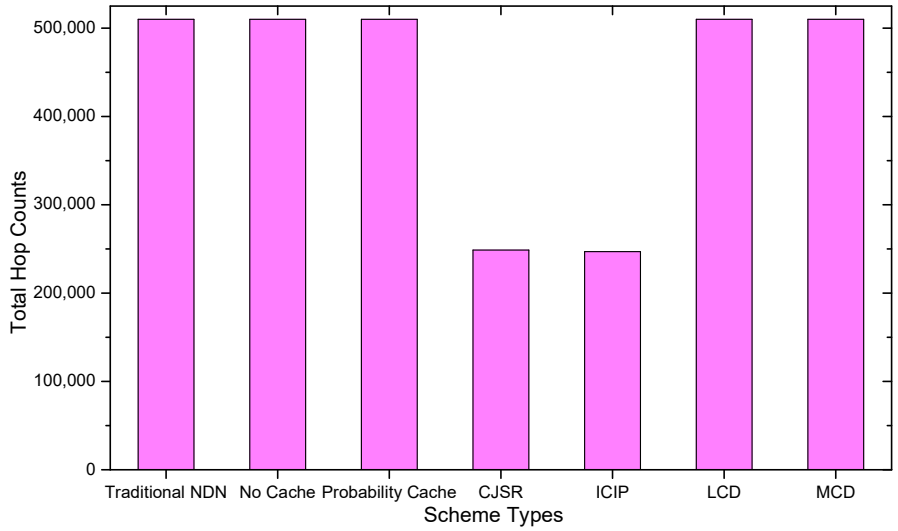

Figure 35. The total hop counts of all the requested data packets in Scenario 6 for User 6.

5.2. Packets Processed Analysis

Figures 36–51 demonstrate the overall traffic of each scheme. The comparisons are made among the number of processed interest packets and data packets of CRs along the caching path. Generally, the traffic burden of the network saw a significant decline after conducting the CJSR scheme. The existing schemes have similar high amount of processed packets, every produced packet has to be processed if there is only one user, and this would inevitably result in high burden on each vertex. Conversely, the processed packets in the proposed schemes are considerably lower, only a small proportion of total produced packets have to be processed. As shown in Figures 36 and 44, the traffic of interest packets had a noticeable drop after conducting pre-caching. The ladder-shaped lines in Figure 36 clearly demonstrate that the vast majority of interest packets are processed by intermediary CRs. The chief reason is that in our schemes, the interest packets only need to flow to the CRs which have cached the requested data. As a result, the introduction of the CJSR scheme could reduce the number of processed interest and data packets on each vertex. Thereafter, the release of traffic burden leads to better User experience. It is also noticeable that the traditional NDN scheme and the Probability Cache scheme nearly perform as bad as No Cache scheme.

In conclusion, vertices that are closer to the user usually had been accessed more frequently in the CJSR scheme and the ICIP scheme, while the burden for the other three schemes is high and remain unchanged as the hop counts get larger. Seen together, the line charts reveal that, in our schemes, the burden of DP or DC was partially taken by CRs. As a result, the burden of dealing with endless packets are reduced and the users get better QoS. The reason is that, in the proposed schemes, interest packets only have to be forwarded to CRs with the corresponding data packets cached. On the contrary, the remaining schemes always have higher number of access because that interest packets have to be forwarded by each CR because data packets are not providently cached for user to fetch.

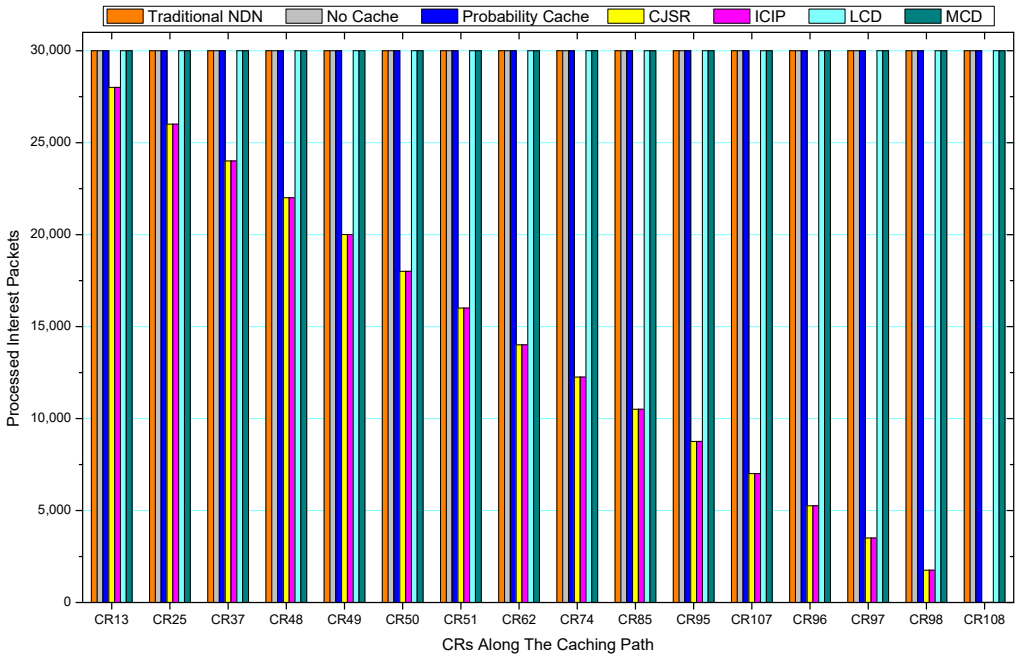

Figure 36. The number of processed interest packets of each CR along Caching Path in Scenario 1.

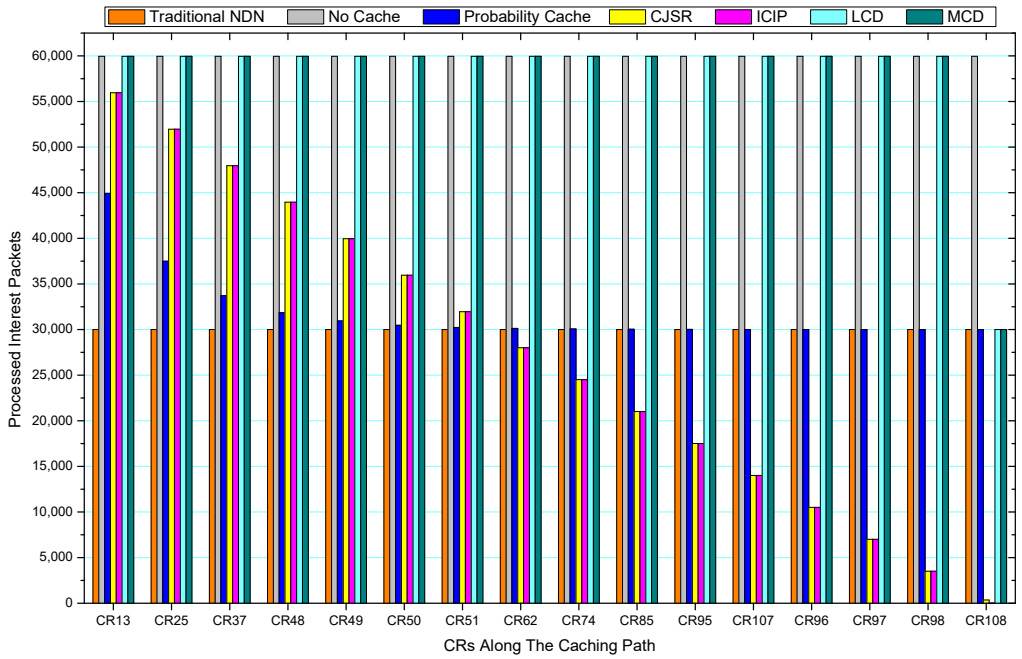

Figure 37. The number of processed interest packets of each CR along Caching Path in Scenario 2.

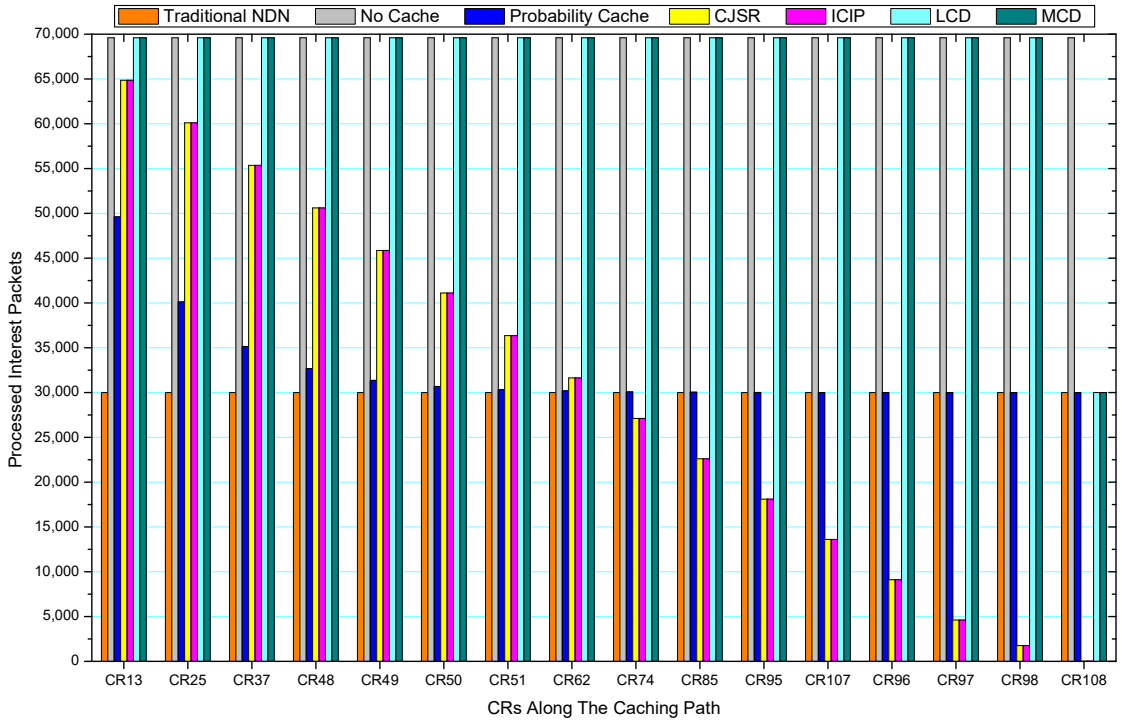

Figure 38. The number of processed interest packets of each CR along Caching Path in Scenario 3.

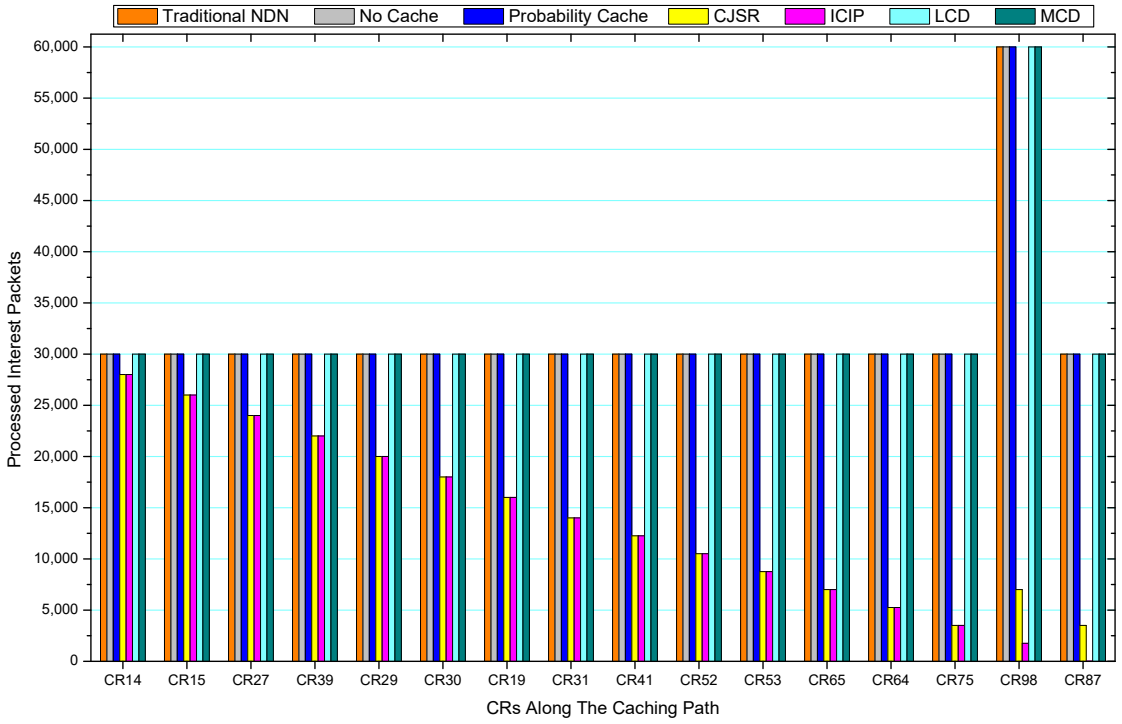

Figure 39. The number of processed interest packets of each CR along Caching Path 1 in Scenario 4.

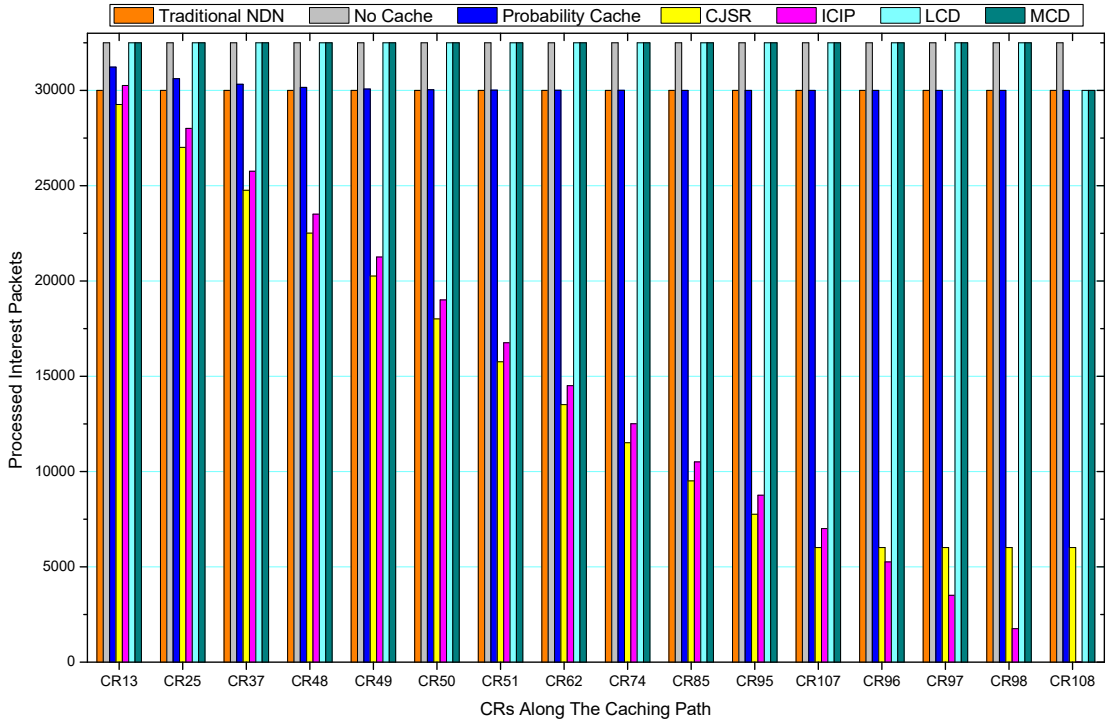

Figure 40. The number of processed interest packets of each CR along Caching Path 1 in Scenario 5.

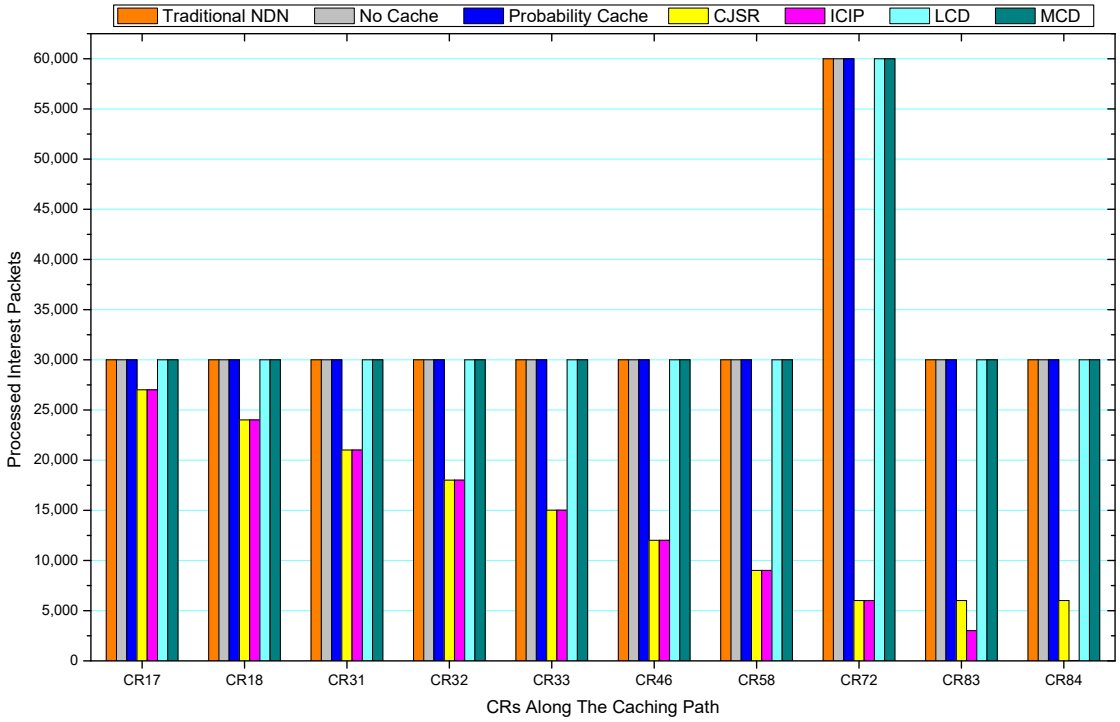

Figure 41. The number of processed interest packets of each CR along Caching Path 3 in Scenario 6.

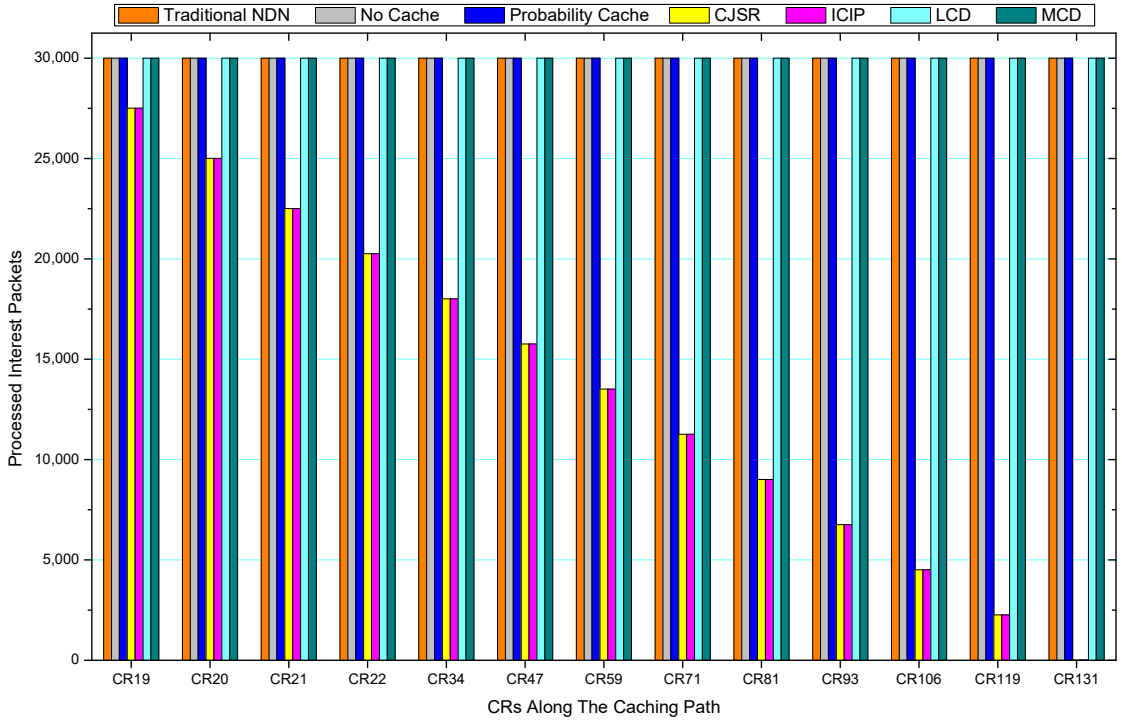

Figure 42. The number of processed interest packets of each CR along Caching Path 4 in Scenario 6.

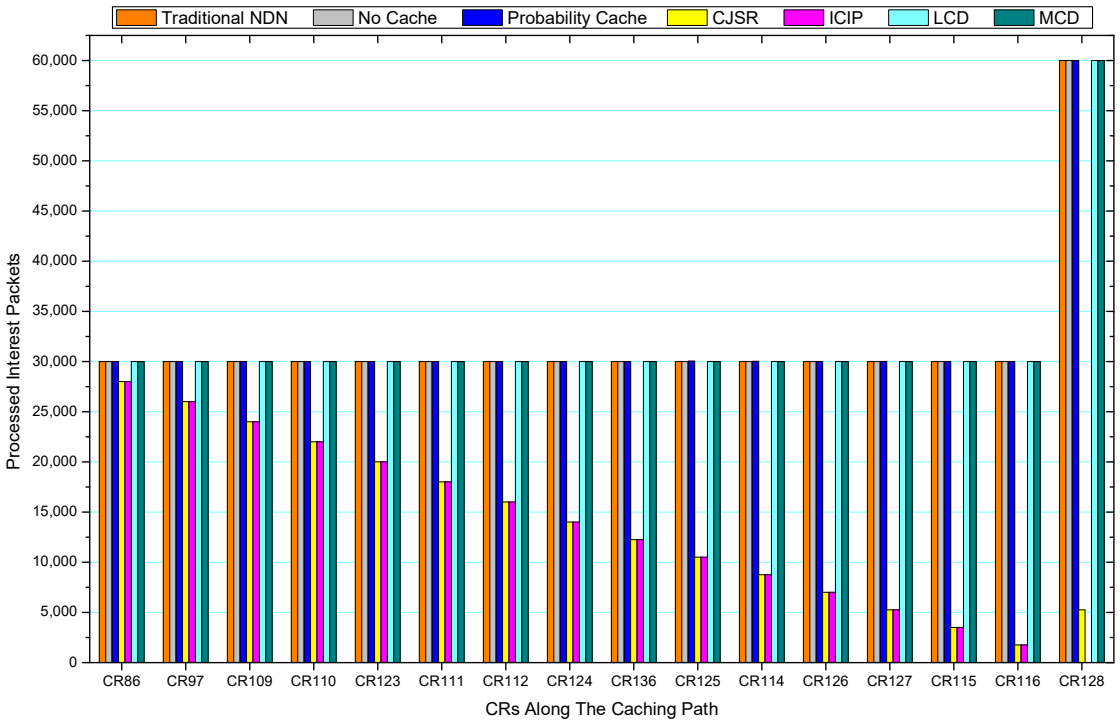

Figure 43. The number of processed interest packets of each CR along Caching Path 6 in Scenario 6.

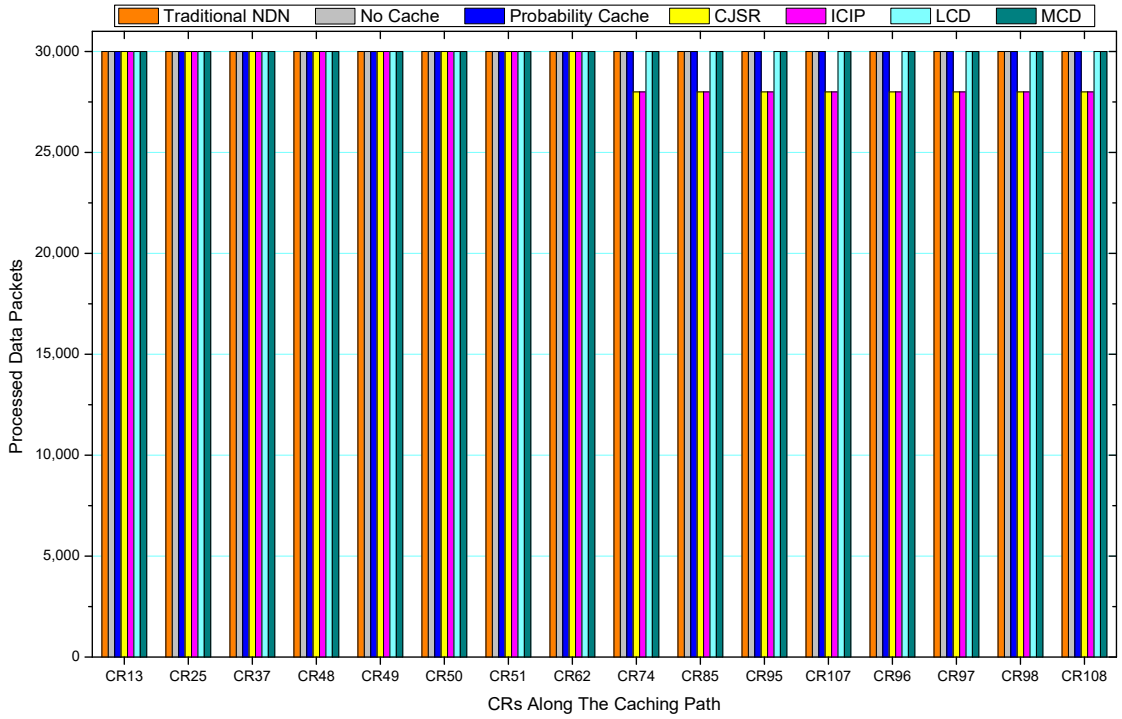

Figure 44. The number of processed data packets of each CR along Caching Path in Scenario 1.

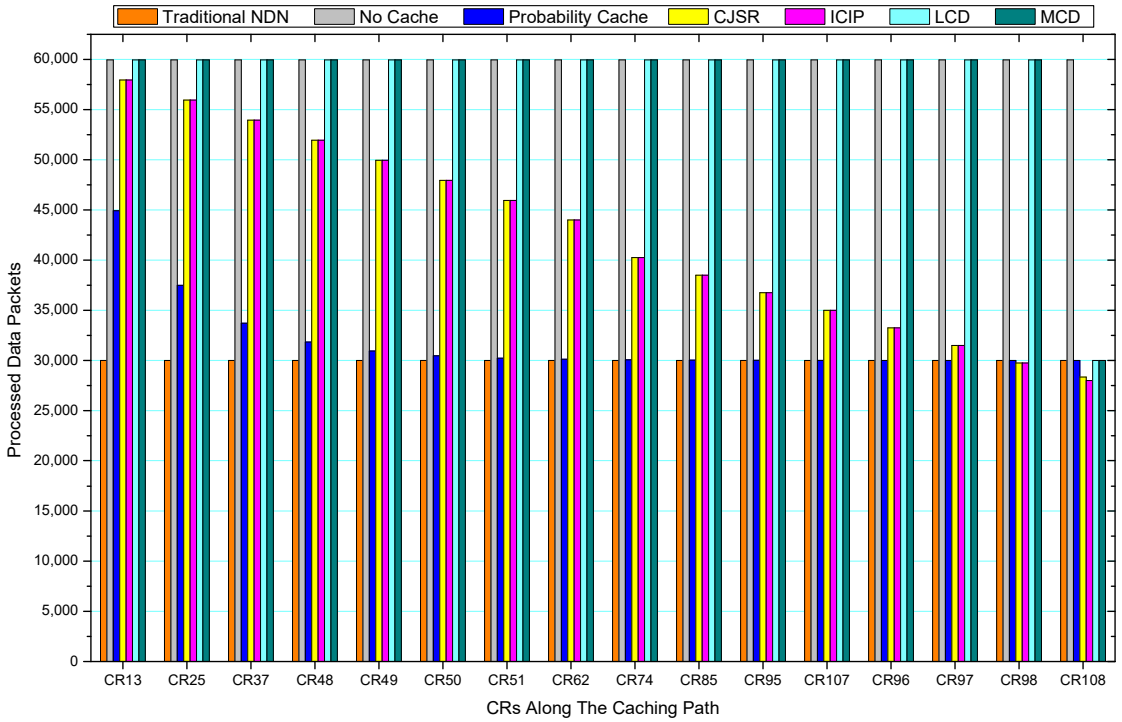

Figure 45. The number of processed data packets of each CR along Caching Path in Scenario 2.

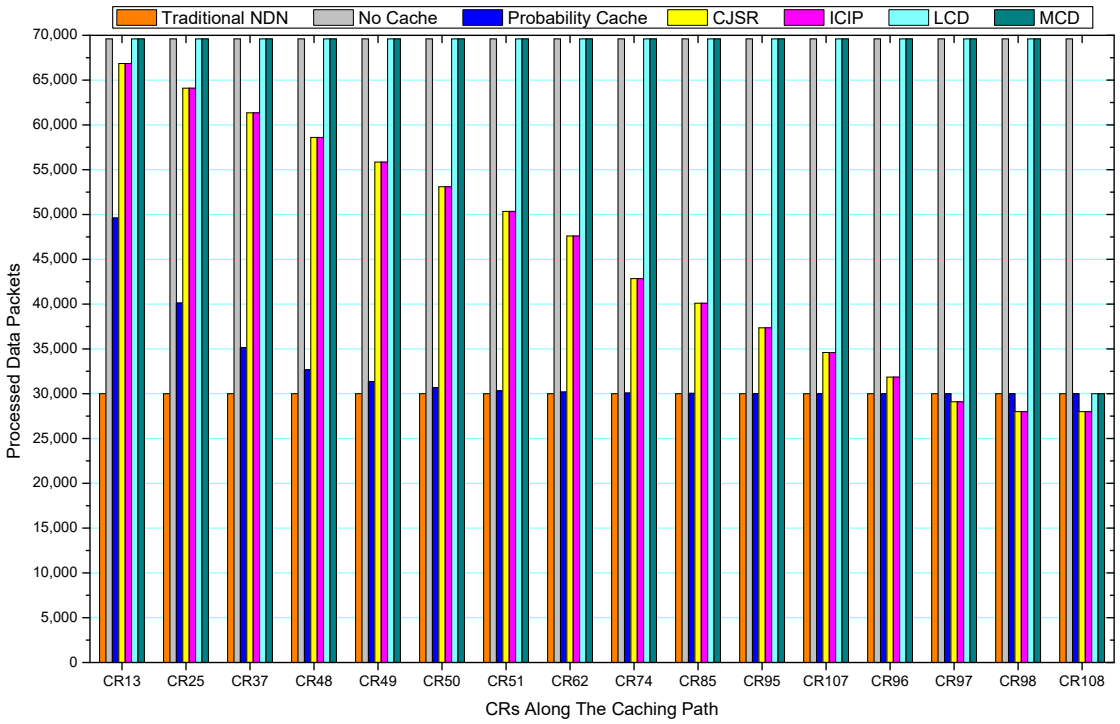

Figure 46. The number of processed data packets of each CR along Caching Path in Scenario 3.

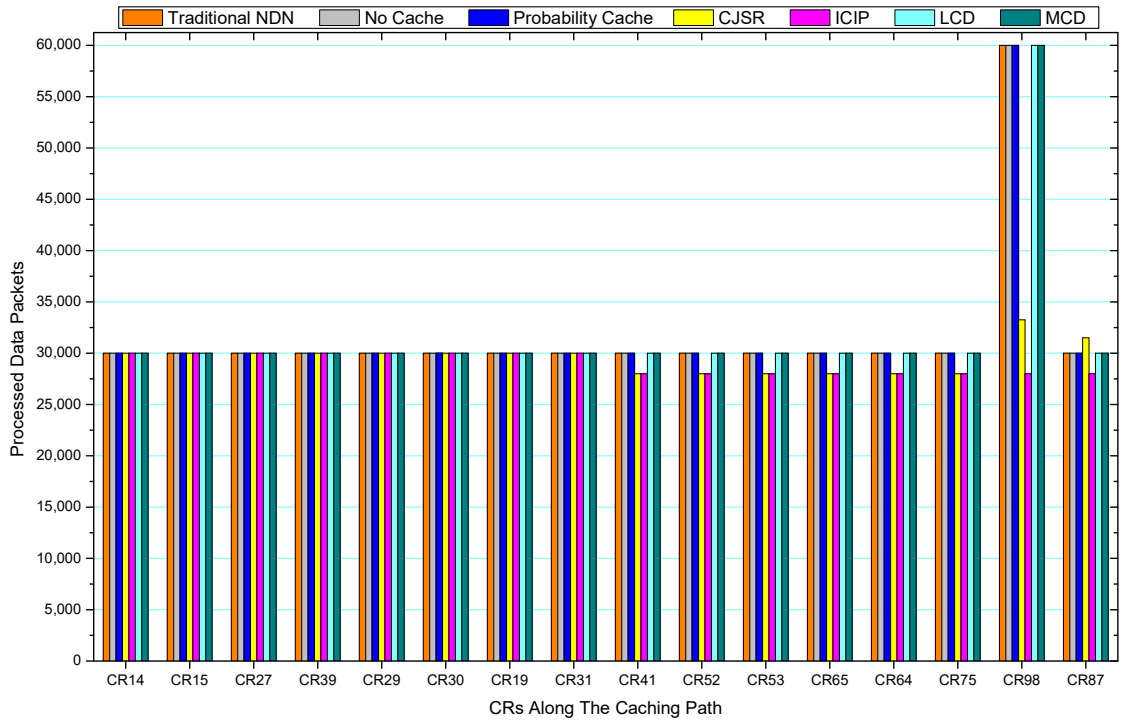

Figure 47. The number of processed data packets of each CR along Caching Path 2 in Scenario 4.

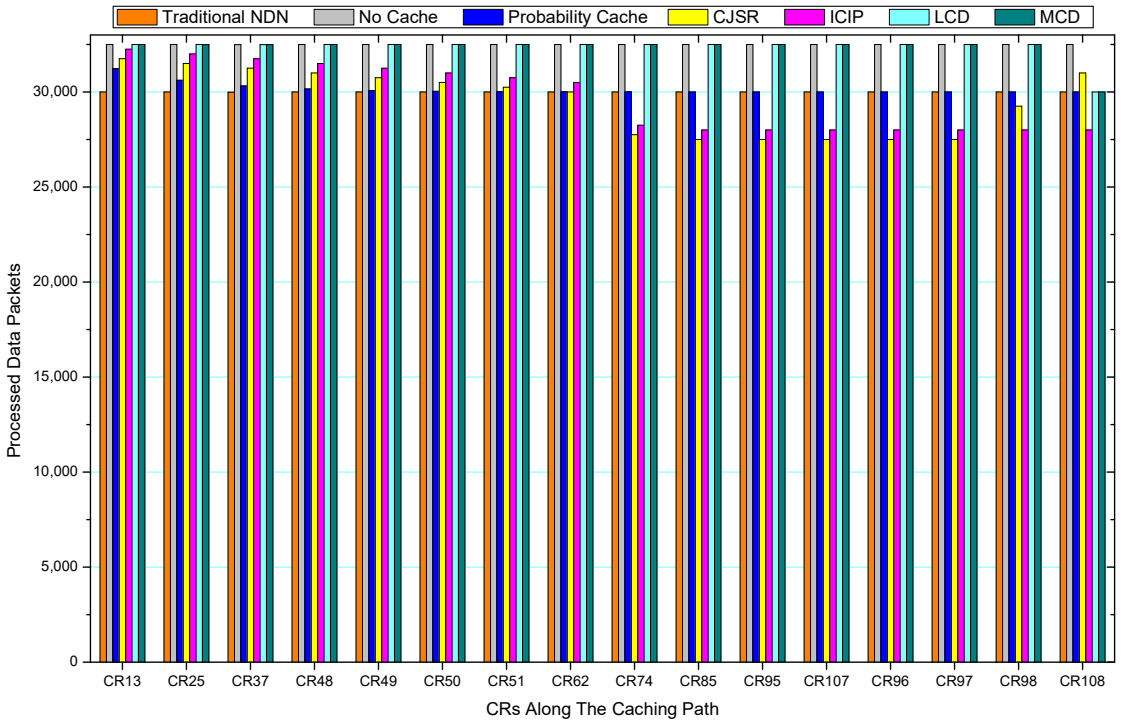

Figure 48. The number of processed data packets of each CR along Caching Path 1 in Scenario 5.

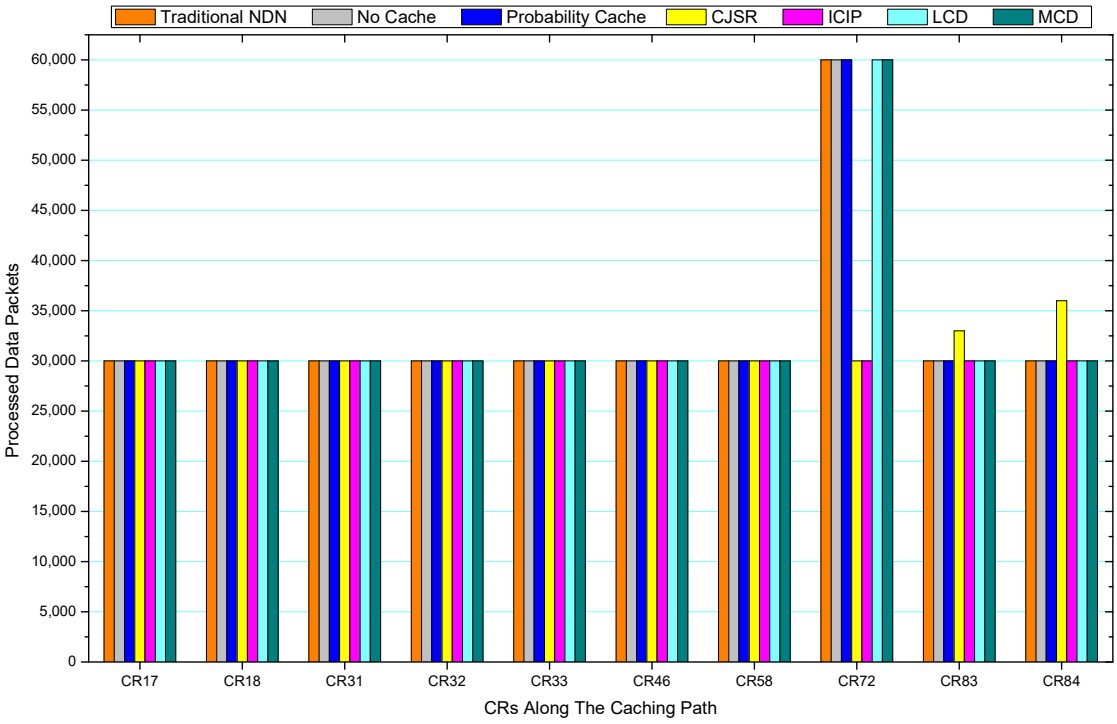

Figure 49. The number of processed data packets of each CR along Caching Path 3 in Scenario 6.

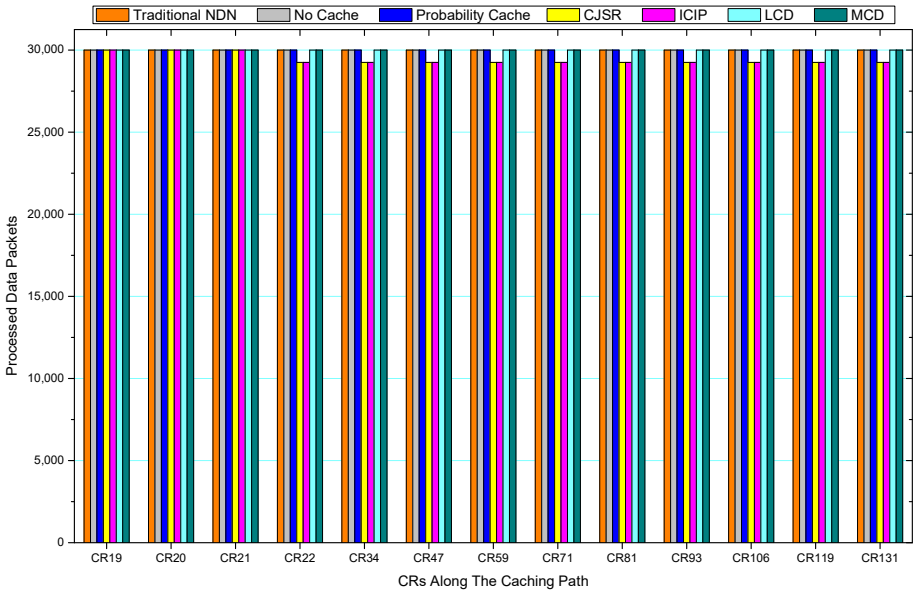

Figure 50. The number of processed data packets of each CR along Caching Path 4 in Scenario 6.

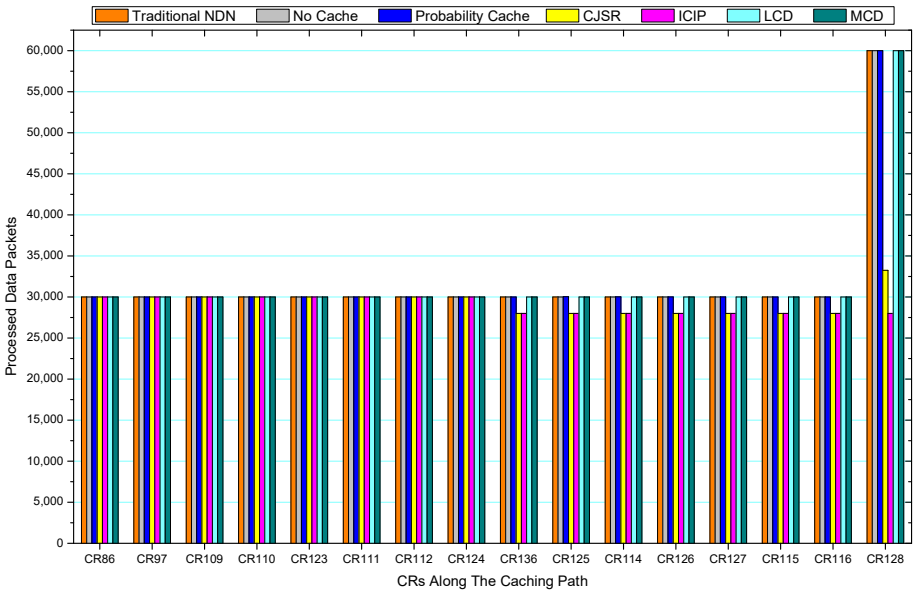

Figure 51. The number of processed data packets of each CR along Caching Path 6 in Scenario 6.

5.3. Cache Hits Analysis

Comparisons between our schemes and the existing caching strategies are depicted in Figures 52–59. Figure 52 demonstrates that proposed schemes' cache hits noticeably outnumber the remaining policies. As illustrated in Figures 53, 54 and 56, all the cache hits of the traditional NDN scheme occurred at ξ_{r_1} (CR 13) and all of cache hits of the LCD and MCD schemes took place at ξ_{p_i} (CR 108). Figures 57–59 also show that all the CRs along the $\mathcal{P}_{cache}$ have high cache hits because latter requests from the same user could benefit from providently cached data. As for the other schemes which have no cache hits at all, each CR only forwards packets. Because data packets are cached to CRs near the users intelligently. Due the involvement of idle CR, the conflicts of data replacement are solved. Thus, cache hits occurred throughout each entire caching path in the ICIP scheme. On the contrary, the DP in the remaining schemes only start to generate data packets when an interest packet is received, then the DP forwards the requested data to users in a time-consuming way.

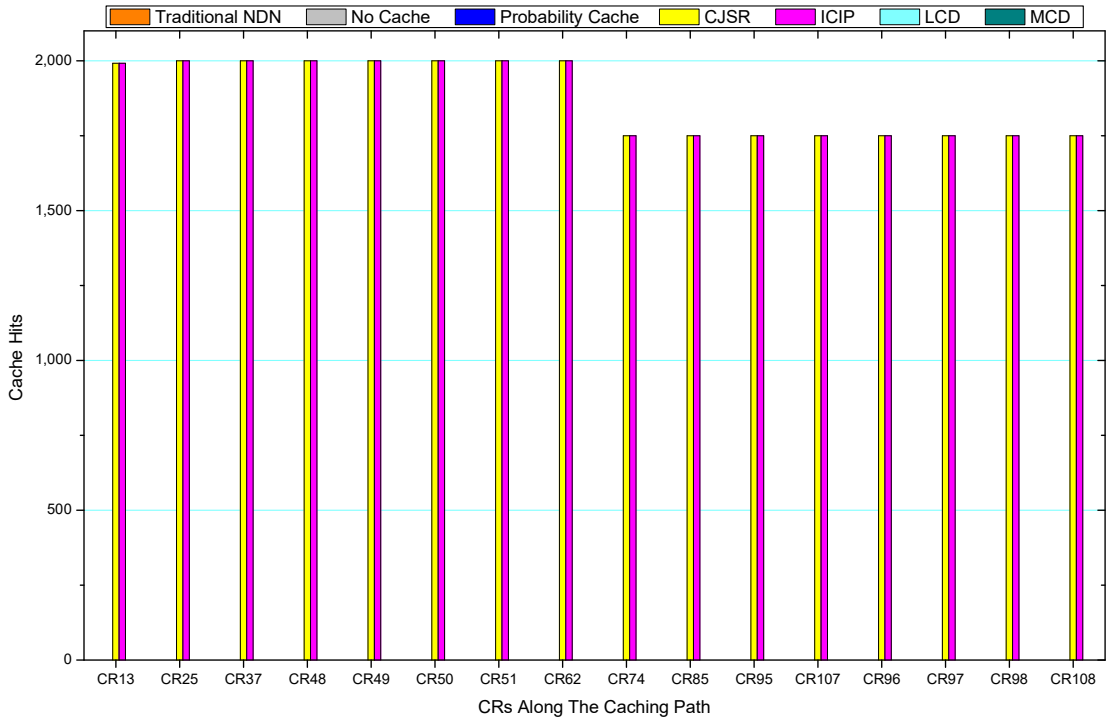

Figure 52. The number of cache hits on each CR along the caching path in Scenario 1.

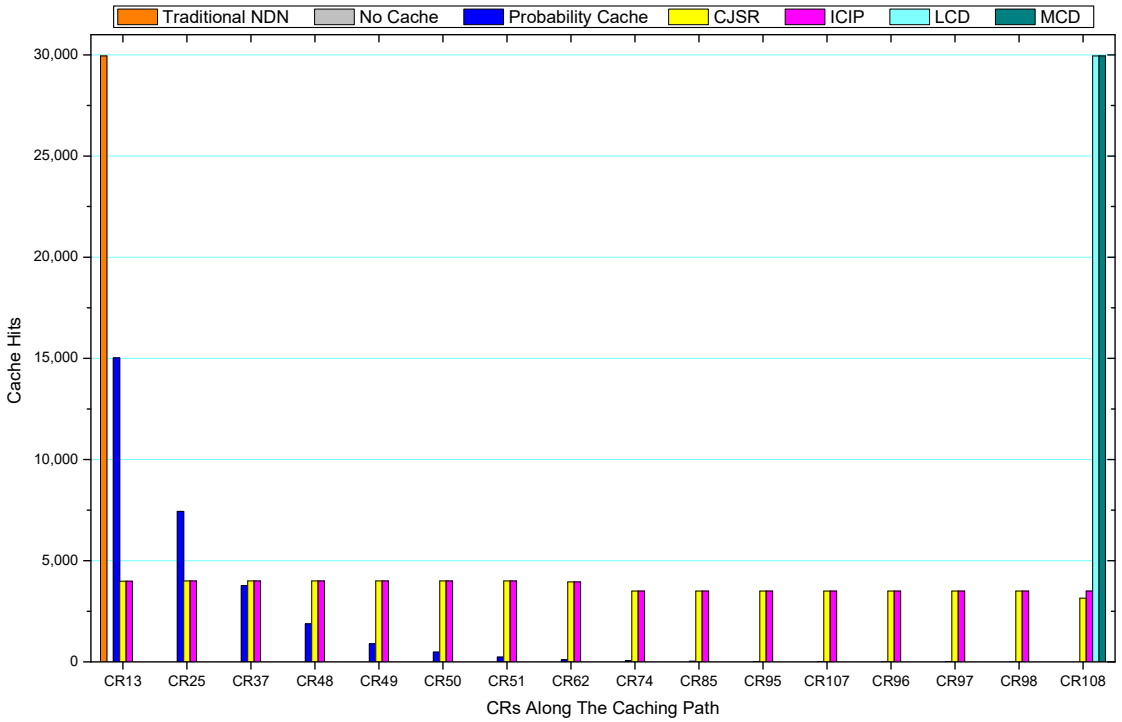

Figure 53. The number of cache hits on each CR along the caching path in Scenario 2.

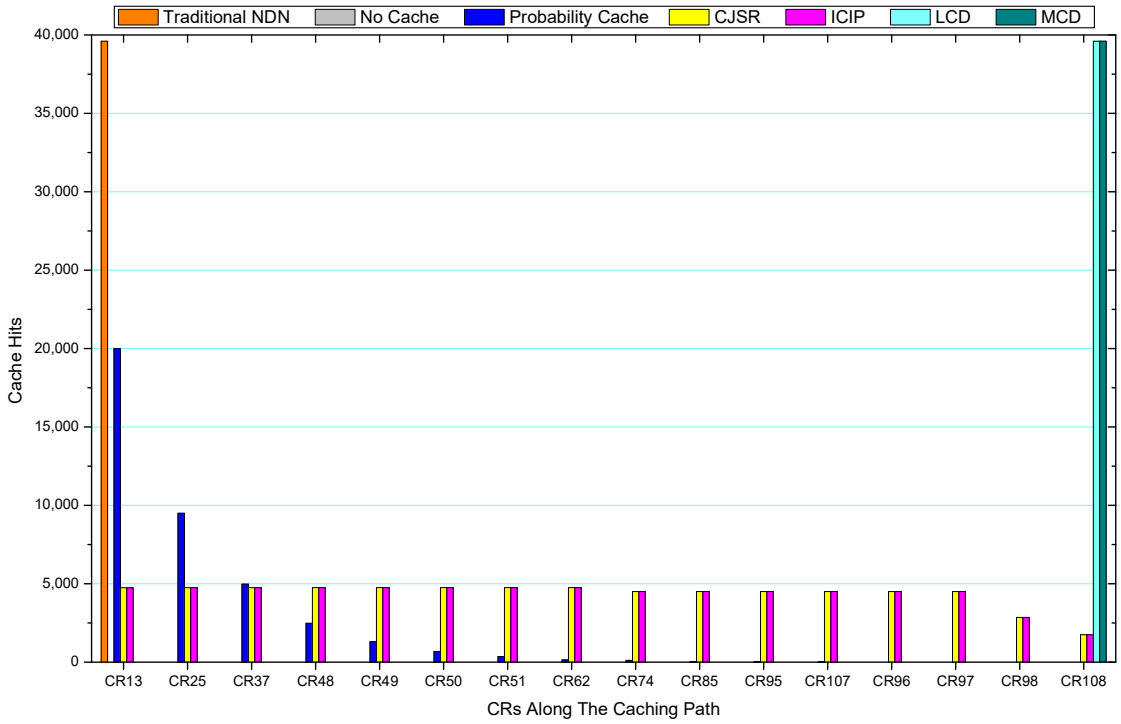

Figure 54. The number of cache hits on each CR along the routing path in Scenario 3.

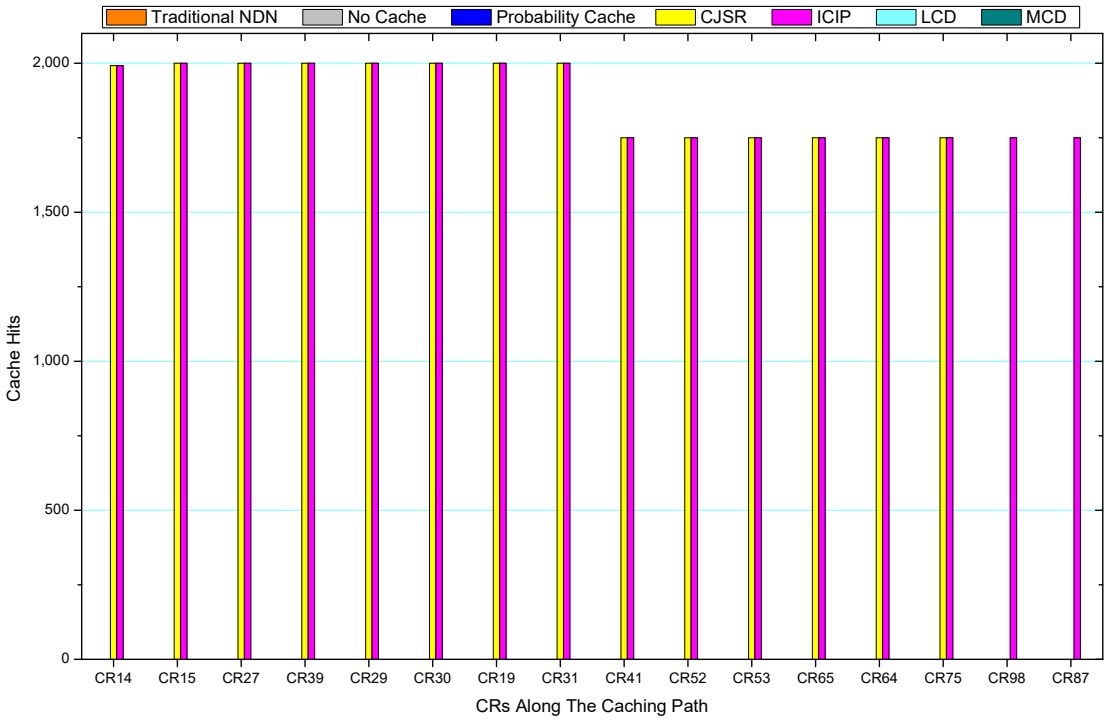

Figure 55. The number of cache hits on each CR along Caching Path 1 in Scenario 4.

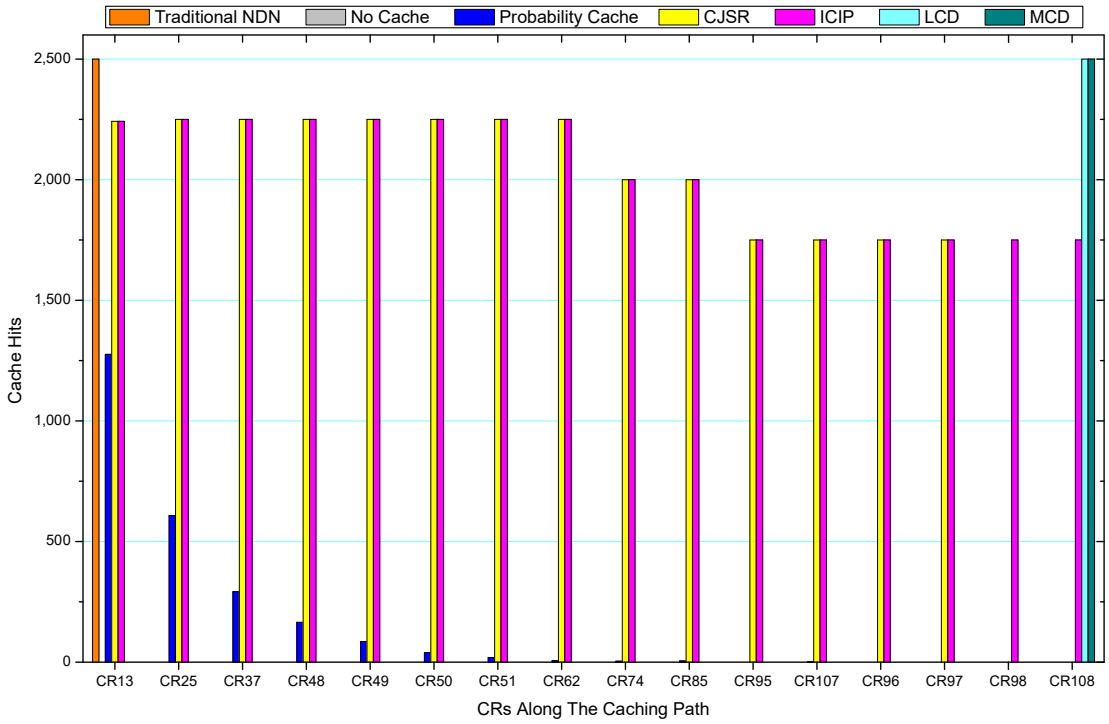

Figure 56. The number of cache hits on each CR along Caching Path 1 in Scenario 5.

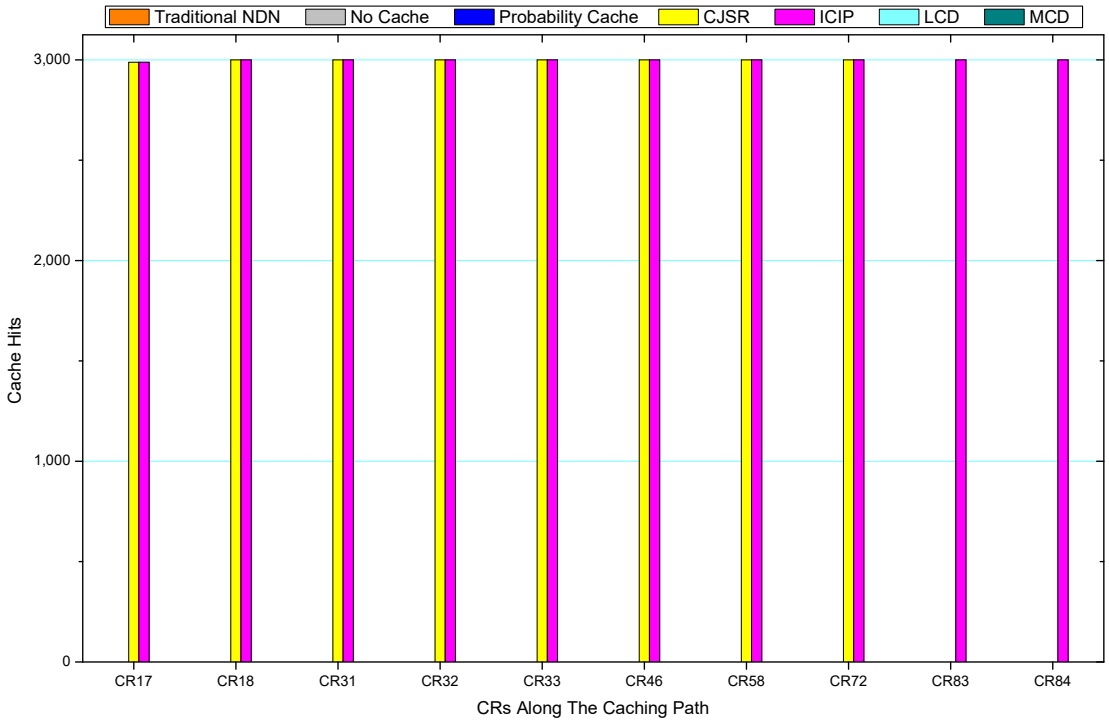

Figure 57. The number of cache hits on each CR along Caching Path 3 in Scenario 6.

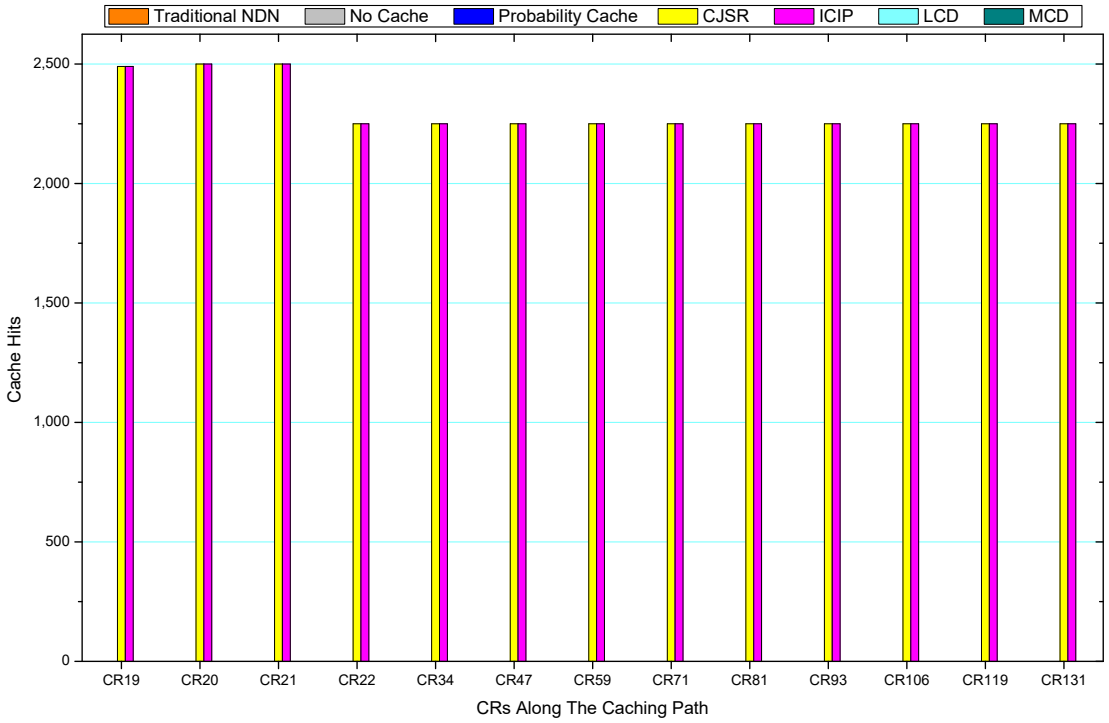

Figure 58. The number of cache hits on each CR along Caching Path 4 in Scenario 6.

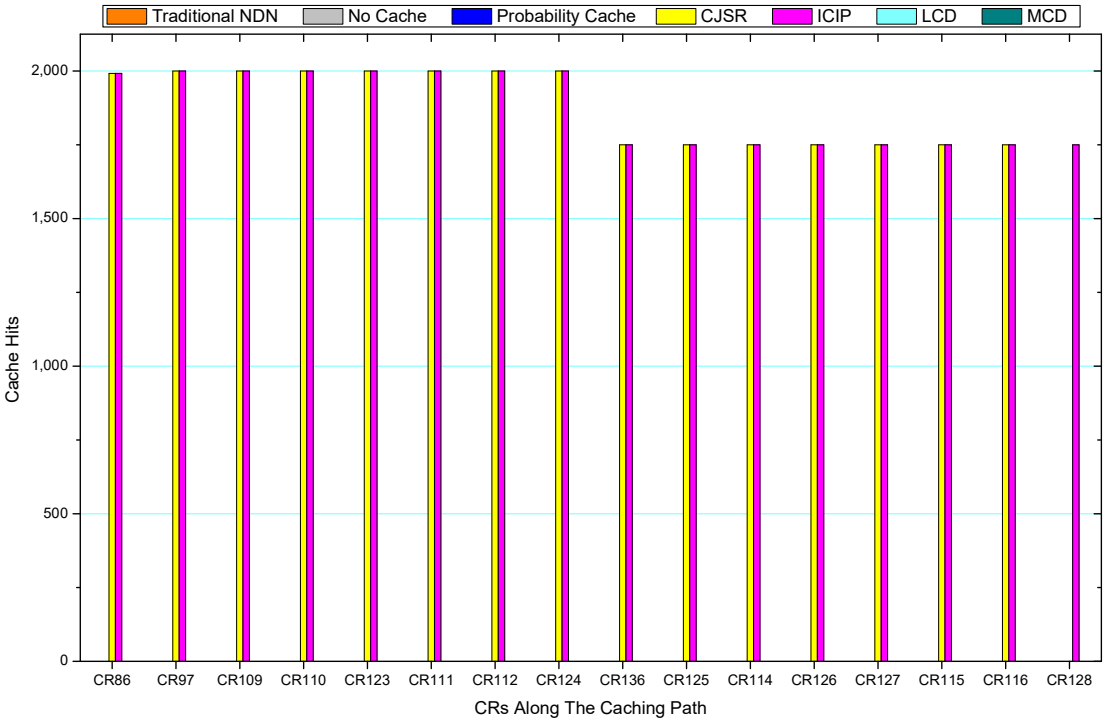

Figure 59. The number of cache hits on each CR along Caching Path 6 in Scenario 6.

6. Conclusions and Future Works

With rapid advances in the manufacturing of sensing devices, the number of data produced by ubiquitous sensing devices grows exponentially and causes heavy burden on the whole network. Aiming to minimize the hop counts for users to obtain data, reduce delay and release traffic burden, a

CJSR scheme is proposed in this paper. The overall length of routing paths is reduced by using two shortcut routing methods. Furthermore, a cooperative pre-caching is proposed to decrease delay and traffic burden. One such scheme is also applicable to different situations. If the requested data have not been uploaded, users would directly request from DPs and trigger uploading pre-caching. If the data have already been uploaded, DCs would perform downloading pre-caching. The cooperative pre-caching would cache data providently into nearby CRs, thus users can directly fetch their interested data from CRs. Since the data can be found in intermediary CRs, the interest packets requesting for data are processed by CRs with the corresponding data cached. Therefore, compared with the traditional NDN scheme, the CJSR scheme could reduce the total number of processed interest packets by 54.8%, enhance the cache hits of each CR and reduce the number of total hop counts by 51.6%. As a result, the traffic burden of both data packets and interest packets is released and the QoS is ameliorated. Using the two methods of shortcut routing, the overall length of routing paths could be cut down by 28.6–85.7%. Moreover, the length of uploading routing path could be decreased by 8.3–33.3%.

With further research on relevant topics and underlying mechanisms involved in the CJSR scheme, namely, replacement approaches and data-freshness, the CJSR scheme could be partially improved. Possible future works include: (i) implementing and evaluating impacts of various replacement policies on the proposed schemes using similar measure as [60]; (ii) designing an adaptive mechanism that could adjust data packets' TTL under various circumstances; and (iii) generating experimental evidence of the impacts on the actions caused by modifying TLV values in messages and improving the statistical analysis under heavier traffic pressure.

Author Contributions: B.H. designed the algorithms and wrote part of the manuscript. A.L. conceived of the work, designed the algorithms, and wrote part of the manuscript. C.Z., N.X., Z.Z. and Z.C. commented on the paper.

Acknowledgments: This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 61572528, 61772554, and 6157256), the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (No. ICT1800391), and the National Basic Research Program of China (973 Program) (Grant No. 2014CB046305).

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Sarkar, S.; Chatterjee, S.; Misra, S. Assessment of the Suitability of Fog Computing in the Context of Internet of Things. *IEEE Trans. Cloud Comput.* **2015**. [CrossRef]
2. Xu, J.; Liu, A.; Xiong, N.; Wang, T.; Zuo, Z. Integrated Collaborative Filtering Recommendation in Social Cyber-Physical Systems. *Int. J. Distrib. Sens. Netw.* **2017**, *13*, 1550147717749745. [CrossRef]
3. Internet of Things Market Forecast: Cisco. Available online: http://postscapes.com/internet-of-things-market-size (accessed on 18 February 2018).
4. Xu, X.; Yuan, M.; Liu, X.; Liu, A.; Xiong, N.; Cai, Z.; Wang, T. Cross-layer Optimized Opportunistic Routing Scheme for Loss-and-delay Sensitive WSNs. *Sensors* **2018**, *18*, 1422. [CrossRef] [PubMed]
5. Kim, H.; Park, J.; Jeong, Y. Sustainable load-balancing scheme for inter-sensor convergence processing of routing cooperation topology. *Sustainability* **2016**, *8*, 436. [CrossRef]
6. Liu, X.; Li, G.; Zhang, S.; Liu, A. Big Program Code Dissemination Scheme for Emergency Software-define Wireless Sensor Networks. *Peer Peer Netw. Appl.* **2017**. [CrossRef]
7. Liu, A.; Chen, W.; Liu, X. Delay Optimal Opportunistic Pipeline Routing Scheme for Cognitive Radio Sensor Networks. *Int. J. Distrib. Sens. Netw.* **2018**. [CrossRef]
8. Xiao, F.; Liu, W.; Li, Z.; Chen, L.; Wang, R. Noise-tolerant Wireless Sensor Networks Localization via Multi-norms Regularized Matrix Completion. *IEEE Trans. Veh. Technol.* **2017**, 1–11. [CrossRef]
9. Zhang, N.; Liang, H.; Cheng, N.; Tang, Y.; Mark, J.W.; Shen, X.S. Dynamic spectrum access in multi-channel cognitive radio networks. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 2053–2064. [CrossRef]

10. Huang, M.; Liu, Y.; Zhang, N.; Xiong, N.; Liu, A.; Zeng, Z.; Song, H. A Services Routing based Caching Scheme for Cloud Assisted CRNs. *IEEE Access* **2018**, *6*, 15787–15805. [CrossRef]
11. Huang, M.; Liu, A.; Zhao, M.; Wang, T. Multi Working Sets Alternate Covering Scheme for Continuous Partial Coverage in WSNs. *Peer Peer Netw. Appl.* **2018**. [CrossRef]
12. Liu, X.; Zhao, S.; Liu, A.; Xiong, N.; Vasilakos, A.V. Knowledge-aware Proactive Nodes Selection Approach for Energy management in Internet of Things. *Future Gen. Comput. Syst.* **2017**. [CrossRef]
13. Liu, X.; Dong, M.; Liu, Y.; Liu, A.; Xiong, N. Construction Low Complexity and Low Delay CDS for Big Data Codes Dissemination. *Complexity* **2018**, *2018*, 5429546. [CrossRef]
14. Lee, J.; Sung, Y.; Park, J. Lightweight sensor authentication scheme for energy efficiency in ubiquitous computing environments. *Sensors* **2016**, *16*, 2044. [CrossRef] [PubMed]
15. Huang, M.; Liu, A.; Xiong, N.; Wang, T.; Vasilakos, A.V. A Low-latency Communication Scheme for Mobile Wireless Sensor Control Systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**. [CrossRef]
16. Wu, M.; Wu, Y.; Liu, C.; Cai, Z.; Xiong, N.; Liu, A.; Ma, M. An Effective Delay Reduction Approach through Portion of Nodes with Larger Duty Cycle for Industrial WSNs. *Sensors* **2018**, *18*, 1535. [CrossRef] [PubMed]
17. Li, J.; Liu, Z.; Chen, X.; Xhafa, F.; Tan, X.; Wong, D.S. L-EncDB: A lightweight framework for privacy-preserving data queries in cloud computing. *Knowl. Based Syst.* **2015**, *79*, 18–26. [CrossRef]
18. Wu, M.; Wu, Y.; Liu, X.; Ma, M.; Liu, A.; Zhao, M. Learning Based Synchronous Approach from Forwarding Nodes to Reduce the Delay for Industrial Internet of Things. *EURASIP J. Wirel. Commun. Netw.* **2018**, *10*. [CrossRef]
19. Liu, X.; Xiong, N.; Zhang, N.; Liu, A.; Shen, H.; Huang, C. A Trust with Abstract Information Verified Routing Scheme for Cyber-physical Network. *IEEE Access* **2018**. [CrossRef]
20. Chen, X.; Li, J.; Weng, J.; Ma, J.; Lou, W. Verifiable computation over large database with incremental updates. *IEEE Trans. Comput.* **2016**, *65*, 3184–3195. [CrossRef]
21. Li, X.; Liu, A.; Xie, M.; Xiong, N.; Zeng, Z.; Cai, Z. Adaptive Aggregation Routing to Reduce Delay for Multi-Layer Wireless Sensor Networks. *Sensors* **2018**, *18*, 1216. [CrossRef] [PubMed]
22. Zhang, Y.; Chen, X.; Li, J.; Wong, D.S.; Li, H.; You, I. Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. *Inf. Sci.* **2017**, *379*, 42–61. [CrossRef]
23. Liu, A.; Zhao, S. High performance target tracking scheme with low prediction precision requirement in WSNs. *Int. J. Ad Hoc Ubiquitous Comput.* **2017**. Available online: http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijahuc (accessed on 3 February 2018).
24. Xu, Q.; Su, Z.; Zheng, Q.; Luo, M.; Dong, B. Secure Content Delivery with Edge Nodes to Save Caching Resources for Mobile Users in Green Cities. *IEEE Trans. Ind. Inform.* **2017**. [CrossRef]
25. Liu, X.; Liu, Y.; Song, H.; Liu, A. Big data orchestration as a service network. *IEEE Commun. Mag.* **2017**, *55*, 94–101. [CrossRef]
26. Liu, Y.; Ota, K.; Zhang, K.; Ma, M.; Xiong, N.; Liu, A.; Long, J. QTSAC: A Energy efficient MAC Protocol for Delay Minimized in Wireless Sensor networks. *IEEE Access* **2018**. [CrossRef]
27. Majeed, M.F.; Ahmed, S.H.; Muhammad, S.; Song, H.; Rawat, D.B. Multimedia streaming in information-centric networking: A survey and future perspectives. *Comput. Netw.* **2017**, *125*, 103–121. [CrossRef]
28. Misra, S.; Chatterjee, S.; Obaidat, M.S. On theoretical modeling of sensor cloud: A paradigm shift from wireless sensor network. *IEEE Syst. J.* **2017**, *11*, 1084–1093. [CrossRef]
29. Liu, Q.; Liu, A. On the hybrid using of unicast-broadcast in wireless sensor networks. *Comput. Electr. Eng.* **2017**. [CrossRef]
30. Yu, S.; Liu, X.; Liu, A.; Xiong, N.; Cai, Z.; Wang, T. Adaption Broadcast Radius based Code Dissemination Scheme for Low Energy Wireless Sensor Networks. *Sensors* **2018**, *18*, 1509. [CrossRef] [PubMed]
31. Bhuiyan, M.Z.A.; Wang, G.; Wu, J.; Cao, J.; Liu, X.; Wang, T. Dependable structural health monitoring using wireless sensor networks. *IEEE Trans. Dependable Secur. Comput.* **2017**, *14*, 363–376. [CrossRef]
32. Li, J.; Chen, X.; Xhafa, F.; Barolli, L. Secure deduplication storage systems supporting keyword search. *J. Comput. Syst. Sci.* **2015**, *81*, 1532–1541. [CrossRef]
33. Liu, A.; Huang, M.; Zhao, M.; Wang, T. A Smart High-Speed Backbone Path Construction Approach for Energy and Delay Optimization in WSNs. *IEEE Access* **2018**. [CrossRef]
34. Neiat, A.G.; Bouguettaya, A.; Sellis, T.; Mistry, S. Crowdsourced Coverage as a Service: Two-Level Composition of Sensor Cloud Services. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 1384–1397. [CrossRef]

35. Majeed, M.F.; Dailey, M.N.; Khan, R.; Tunpan, A. Pre-caching: A proactive scheme for caching video traffic in named data mesh networks. *J. Netw. Comput. Appl.* **2017**, *87*, 116–130. [CrossRef]
36. Liu, Y.; Liu, A.; Guo, S.; Li, Z.; Choi, Y.J. Context-aware collect data with energy efficient in Cyber-physical cloud systems. *Future Gen. Comput. Syst.* **2017**. [CrossRef]
37. Jiang, W.; Wang, G.; Bhuiyan, M.Z.A.; Wu, J. Understanding graph-based trust evaluation in online social networks: Methodologies and challenges. *ACM Comput. Surv.* **2016**, *49*, 10. [CrossRef]
38. Xu, Q.; Su, Z.; Yang, K. Optimal Control Theory-Based Epidemic Information Spreading Scheme for Mobile Social Users with Energy Constraint. *IEEE Access* **2017**, *5*, 14107–14118. [CrossRef]
39. Tang, J.; Liu, A.; Zhang, J.; Zeng, Z.; Xiong, N.; Wang, T. A Security Routing Scheme Using Traceback Approach for Energy Harvesting Sensor Networks. *Sensors* **2018**, *18*, 751. [CrossRef] [PubMed]
40. Zhu, H.; Xiao, F.; Sun, L.; Wang, R.; Yang, P. R-TTWD: Robust device-free through-the-wall detection of moving human with WiFi. *IEEE J. Sel. Areas Commun.* **2017**, *35*, 1090–1103. [CrossRef]
41. Wang, J.; Liu, A.; Yan, T.; Zeng, Z. A Resource Allocation Model Based on Double-sided Combinational Auctions for Transparent Computing. *Peer Peer Netw. Appl.* **2017**. [CrossRef]
42. Xin, H.; Liu, X. Energy-balanced transmission with accurate distances for strip-based wireless sensor networks. *IEEE Access* **2017**, *5*, 16193–16204. [CrossRef]
43. Gui, J.; Deng, J. Multi-hop Relay-Aided Underlay D2D Communications for Improving Cellular Coverage Quality. *IEEE Access* **2018**, *6*, 14318–14338. [CrossRef]
44. Nguyen, P.L.; Ji, Y.; Liu, Z.; Vu, H.; Nguyen, K.V. Distributed hole-bypassing protocol in WSNs with constant stretch and load balancing. *Comput. Netw.* **2017**, *129*, 232–250.
45. Chen, X.; Pu, L.; Gao, L.; Wu, W.; Wu, D. Exploiting massive D2D collaboration for energy-efficient mobile edge computing. *IEEE Wirel. Commun.* **2017**, *24*, 64–71. [CrossRef]
46. Pu, L.; Chen, X.; Xu, J.; Fu, X. D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3887–3901. [CrossRef]
47. Søgaard, J.; Shahid, M.; Pokhrel, J.; Brunnström, K. On subjective quality assessment of adaptive video streaming via crowdsourcing and laboratory based experiments. *Multimedia Tools Appl.* **2017**, *76*, 16727–16748. [CrossRef]
48. Rahmani, A.M.; Gia, T.N.; Negash, B.; Anzanpour, A.; Azimi, I.; Jiang, M.; Liljeberg, P. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Gen. Comput. Syst.* **2018**, *78*, 641–658. [CrossRef]
49. Li, Z.; Chang, B.; Wang, S.; Liu, A.; Zeng, F.; Luo, G. Dynamic Compressive Wide-band Spectrum Sensing Based on Channel Energy Reconstruction in Cognitive Internet of Things. *IEEE Trans. Ind. Inform.* **2018**. [CrossRef]
50. Bhuiyan, M.Z.A.; Wu, J.; Wang, G.; Wang, T.; Hassan, M.M. e-Sampling: Event-Sensitive Autonomous Adaptive Sensing and Low-Cost Monitoring in Networked Sensing Systems. *ACM Trans. Auton. Adapt. Syst.* **2017**, *12*, 1. [CrossRef]
51. Chiocchetti, R.; Perino, D.; Carofiglio, G.; Rossi, D.; Rossini, G. INFORM: A dynamic Interest Forwarding Mechanism for Information Centric Networking. In Proceedings of the ACM SIGCOMM Workshop Information-Centric Networking, Hong Kong, China, 12 August 2013; pp. 9–14.
52. Islam, S.M.; Moon, A.R. *Analysis of LCD (Leave Copy Down) &LCE (Leave Copy Everywhere) Caching Scheme for Tree Topology*; East West University: Chicago, IL, USA, 2016; pp. 12–15.
53. Laoutaris, N.; Che, H.; Stavrakakis, I. The LCD interconnection of LRU caches and its analysis. *Perform. Eval.* **2006**, *63*, 609–634. [CrossRef]
54. Chai, W.K.; He, D.; Psaras, I.; Pavlou, G. Cache "less for more" in information-centric networks. In Proceedings of the International Conference on Research in Networking, Prague, Czech Republic, 21–25 May 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 27–40.
55. Psaras, I.; Chai, W.K.; Pavlou, G. Probabilistic in-network caching for information-centric networks. In Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking, Helsinki, Finland, 17 August 2012; pp. 55–60.
56. Ming, Z.X.; Xu, M.W.; Wang, D. Age-Based cooperative caching in information-centric networks. In Proceedings of the 2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Orlando, FL, USA, 25–30 March 2012; pp. 268–273.

57. Cho, K.; Lee, M.; Park, K.; Kwon, T.T.; Choi, Y.; Pack, S. WAVE: Popularity-based and collaborative in-network caching for content-oriented Networks. In Proceedings of the 2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Orlando, FL, USA, 25–30 March 2012; pp. 316–321.
58. Zhang, G.Q.; Li, Y.; Lin, T. Caching in information centric networking: A survey. *Comput. Netw.* **2013**, *57*, 3128–3141. [CrossRef]
59. Guo, Y.; Liu, F.; Cai, Z.; Xiao, N.; Zhao, Z. Edge-Based Efficient Search over Encrypted Data Mobile Cloud Storage. *Sensors* **2018**, *18*, 1189. [CrossRef] [PubMed]
60. Panagiotou, C.; Antonopoulos, C.; Koubias, S. Performance enhancement in WSN through data cache replacement policies. In Proceedings of the 2012 IEEE 17th International Conference on Emerging Technologies and Factory Automation, ETFA, Krakow, Poland, 17–21 September 2012; pp. 1–8. [CrossRef]