



Article

A Periodic Caching Strategy Solution for the Smart City in Information-Centric Internet of Things

Muhammad Ali Naeem ¹, Rashid Ali ² , Byung-Seo Kim ^{3,*} , Shahrudin Awang Nor ¹ and Suhaidi Hassan ¹

¹ InterNetworks Research Laboratory School of Computing (SOC), University Utara Malaysia, 06010 UUM Sintok, Malaysia; malinaeem7@gmail.com (M.A.N.); shah@uum.edu.my (S.A.N.); suhaidi@uum.edu.my (S.H.)

² Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Korea; rashid@ynu.ac.kr

³ Department of Software and Communications Engineering, Hongik University, Sejong 30016, Korea

* Correspondence: jsnbs@hongik.ac.kr; Tel.: +82-41-860-2539

Received: 17 June 2018; Accepted: 20 July 2018; Published: 23 July 2018



Abstract: Named Data Networking is an evolving network model of the Information-centric networking (ICN) paradigm which provides Named-based data contents. In-network caching is the responsible for dissemination of these contents in a scalable and cost-efficient way. Due to the rapid expansion of Internet of Things (IoT) traffic, ICN is envisioned to be an appropriate architecture to maintain the IoT networks. In fact, ICN offers unique naming, multicast communications and, most beneficially, in-network caching that minimizes the response latency and server load. IoT environment involves a study of ICN caching policies in terms of content placement strategies. This paper addressed the caching strategies with the aim to recognize which caching strategy is the most suitable for IoT networks. Simulation results show the impact of different IoT ICN-based caching strategies, out of these; periodic caching is the most appropriate strategy for IoT environments in terms of stretch that results in decreasing the retrieval latency and improves the cache-hit ratio.

Keywords: Internet of Things; named data networking; caching; most popular cache; 5G

1. Introduction

Internet of Things (IoT) is a discreet revolution that is aiming to change the future of the present world. Nowadays, the connected world is becoming a reality [1]. This promising field of research encompasses several domains, including the most valuable areas, such as smart wearables, smart homes, smart cities, smart agriculture, smart healthcare systems, smart industries, smart energy management, and smart transportation systems. Therefore, it can be said that IoT is an emerging network of miscellaneous resources (memory and power) constraining physical devices [2]. On the other hand, Named Data Networking (NDN) is an innovative paradigm that provides the location independent data communications in which client's requests are satisfied regardless of the content's location. In fact, this concept provides unique and location-independent content, in-network caching and name-based routing. These features give NDN the potential to become a key technology for data dissemination in IoT. The NDN concept has abilities to mitigate the significant communication overhead caused by the distribution of commonly accessed contents.

In recent years, IoT has grasped researchers' interest to improve the efficiency of data transmission. The exponential increase in the number of connected things (devices) through the Internet and the progress in wireless communication technologies have maximized network data traffic. In fact, recent transmitting data traffic statistics published by Cisco agree, showing that annual Internet

protocol (IP) traffic around the globe has a threshold of more than 1 zettabyte (ZB) (1 ZB = 1000 exabytes) in 2018, and expectedly, it will reach 2.3 ZB in 2020 [3]. Internet usage has increased extensively in the last decade; video on demand (VoD) is especially widely used, increasing Internet traffic. The currently working Internet architecture supports end-to-end communication, and the content retrieval process is presented in two steps. First, the name resolution provides identifiers to the contents (i.e., uniform resource locator [URL]) with a location (i.e., IP address) and second, there is a location-based transmission for user requests (from user to source) and requested contents (from source to user). The name resolution is provided to the contents at the application layer before forwarding the contents that waste the resources of the network [4]. For instance, if a copy of the requested content resides near the user but in the IP network, the user's request needs to be transmitted to the server and it will increase traffic and the use of network resources [5]. Moreover, users want their desired content regardless of having information about the content location. Due to such situations, the network design needs to be changed. In this case, information-centric networking (ICN) [5] is a new paradigm that plays an important role because of its location-independent architectural design. It provides a named content-based approach regardless of the content's physical location, and subscribers' requests are satisfied by the nearest replica. ICN is a promising development for different IoT environments [6]. Consequently, ICN delivers location-independent data dissemination through unique content names and name-based routing, and it is the most advantageous form of in-network caching. These ICN features give it the potential to become crucial for data distribution in IoT. Certainly, it provides claim data access and diminishes retrieval delay and the burden on the publisher.

The ICN conception is to decrease the substantial communication overhead caused by the dissemination of analogously accessed contents. Users' requests are rarely contented by the main publishers. On the other hand, in-networking caching arranges a content replica that is cached at different locations to increase the availability of the desired data. Caching is a crucial component that makes ICN more beneficial than the present Internet architecture. It significantly minimizes the huge volume of data traffic and diminishes the bottleneck caused by publishing data in a unique location. Moreover, caches reduce the distance required for data retrieval. The cache performance totally depends on cache management schemes, which are divided into two mechanisms, namely, content placement strategies and replacement policies. Placement strategies are responsible for tracing the location for disseminated content to be cached according to the placement algorithm, while replacement policies are needed to make room for new incoming contents, once the caches overflow. In fact, subscribers seek data through connecting with different mobile or fixed devices that belong to diverse environments. As a result, caching content within the network will improve the lifetime of the IoT devices' batteries; for example, an active node may fulfill a request while the information producer remains in sleep mode. Some well-known cache management strategies are Leave Copy Everywhere (LCE) [7], which caches the content at all on-path routers, and Leave Copy Down (LCD) [8], which performs caching at a one level downstream router and edge caching [9] and it copies requested content at only one router according to the stretch. Moreover, ProbCache [10] caches content according to free cache space. The objective of these in-network caching strategies is to improve the overall network performance. Therefore, these strategies focus on crucial issues, such as network bandwidth, power consumption, caching content near consumers, distance between the user and the source, resource (cache and batteries) usability, and content diversity, to increase the availability of the desired information. In such a situation, popularity-based caching strategies achieve a better performance [5,11].

This paper addressed the caching strategies facing several challenges (e.g., Latency, stretch, data availability) with the aim of recognizing which caching strategy is the most suitable for IoT networks. We performed simulations that indicated the impact of different popularity-based caching strategies; of these, periodic caching is the most appropriate strategy for IoT environments in terms of a stretch that results in decreasing the content retrieval latency and improving the cache-to-hit ratio [12]. Moreover, it is useful for all smart city applications, as well.

The remainder of the paper is organized as follows. Section 2 describes an overview of Named Data Networking (NDN). In Section 3, the related research work is described in detail. A periodic caching strategy (PCS) is proposed in Section 4. Section 5 describes a performance evaluation of the proposed caching strategy in comparison to state-of-the-art strategies. Finally, in Section 6, a comprehensive conclusion is presented.

2. Overview of Named Data Networking

NDN [13] is the first information-centric architecture that was implemented at Palo Alto Research Center (PARC), and it is one of the most attractive ICN-based architectures [12]. The goal of NDN is to change host-based Internet into named content-based Internet. NDN is one of the most significant research fields for designing the Future Internet [3]. NDN has been divided into several modules: naming, routing, security, mobility, and caching. Each module has its own properties that provide benefits to all NDN architectures. In NDN, data are presented in the form of data contents that are associated with a specific naming structure (hierarchical and flat). With the help of two primitives called the interest packet and the data content, dissemination is fabricated in NDN. The interest packet is used to denote the user's requests and the data content shows the response from the source [3].

When a user sends his or her interest packet (request) for some data contents, then the interest packet matches all the on-path routers with existing data contents in the form of a prefix order. An NDN router is made up of three parts, i.e., the content store (CS), pending interest table (PIT), and forwarding information base (FIB). The CS has the ability to store incoming contents, as well as users' requests at the data delivery path. The PIT manages all the requests received from consumers, and it specifies an appropriate path for incoming requests [14]. Meanwhile, the FIB provides information about topology for the interest packet and data contents to move to the suitable publisher and consumers. A working NDN architecture is illustrated in Figure 1. When a consumer sends a request to the NDN network, the request is then transmitted to the nearest router, and it is compared with the content stored in the CS [15]. If the desired content is found there, then the content is sent to the consumer instantly through the return path, and a copy of the requested content is cached at all the routers alongside the data delivery path. If the required content does not match any of the existing data entries, then the CS forwards the request to the appropriate source [4].

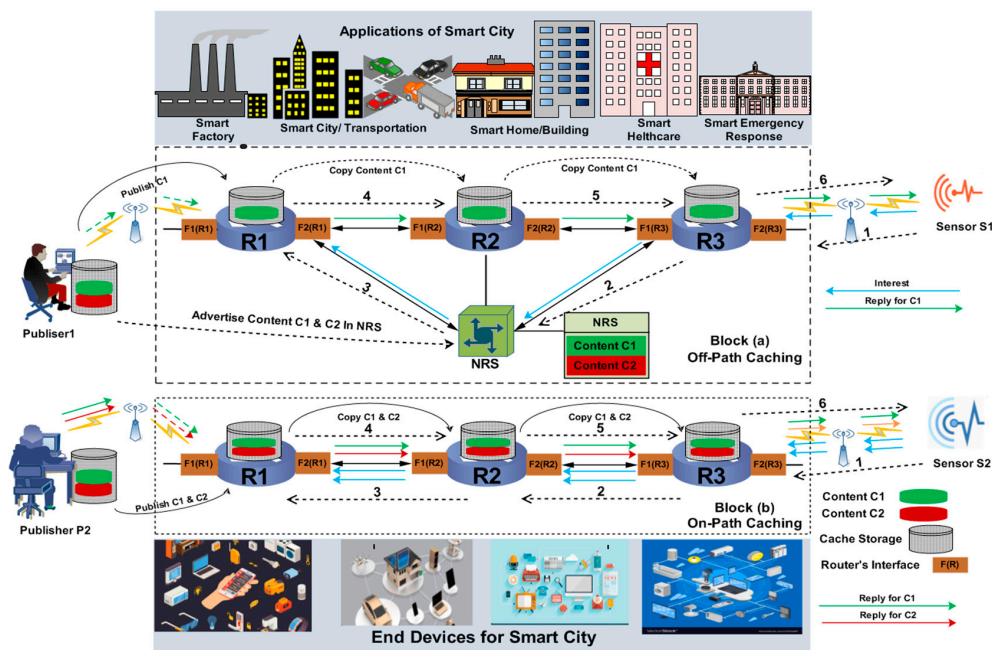


Figure 1. Off-path and On-path Caching mechanism in NDN.

2.1. Cache in Named Data Networking

The incredible growth of today's Internet traffic and repeated transmissions of the same contents have created many problems, particularly in terms of bandwidth utilization, server load, response time, and usage of resources. These growing problems will be difficult to solve in the future with an IP-based Internet architecture [3]. As we know, Internet use is increasing day by day, and current resources are insufficient to overcome the huge number of ever-increasing needs of the Internet users [16]. To achieve a better performance, the Internet needs to implement an optimal approach that can face these increasing difficulties [17]. In this case, ICN provides a better solution to resolve these expected issues by implementing in-network caching, which is a basic and unavoidable module in all ICN architectures. Caching is temporary storage used to speed up the communication process [18]. Essentially, it is used to minimize the average cost (time and energy) and improve the content retrieval process, because a cache stores temporarily a copy of frequently requested data. Web caching is a famous example of this temporary storage; it involves storing web documents (e.g., HTML page images) [19] to reduce bandwidth consumption, server load, and response time. Caching makes ICN differ from the presently running IP-based Internet, as caching is used to transmit content near users, which decreases response time and resource utilization and improves data transmission services. In Figure 1, a user requests a specific piece of information, which can be retrieved by any router having the corresponding content in its cache [20].

The main problem is the capacity of in-network caches, which provide much smaller space to accommodate huge data transmissions. Therefore, efficient content distribution can be achieved by integrating a significant number of caches, but this is not easy. Effective content distribution requires a scheme design that will disseminate the contents in a well-organized manner with fewer resources (caches) used. It can be handled using an appropriate cache management strategy. For this reason, several caching strategies are used to cache requested contents near users for subsequent requests to reduce the source and network load [21].

According to the caching criteria, different types of caching strategies have been recently proposed to achieve a better caching performance [22]. For example, caching decision is based on the distance from the source to the user, centrality, probabilistic logic, content popularity, and label-based caching, and it can be done by adopting a mathematical equation. These strategies boost information replication and decrease the reaction time [23]. In addition, they reduce the number of network resources and minimize network traffic [20]. Caching strategies are used to improve the content delivery service with the efficient use of memory [24]. NDN caching is divided into two categories: off-path caching and on-path caching. Figure 1 demonstrates these caching mechanisms.

2.2. Off-Path Caching

Off-path caching is akin to traditional proxy caching or a content delivery network (CDN) server. Off-path caching requires integrating a special entity called a name resolution system (NRS). All contents are registered in the NRS at the time of creation, and consumers can send their requests for registered contents [25]. Exact information about the contents depends on the NRS, but the most necessary step is to keep updated about local content information. In Figure 1 (block a), a request is sent from Sensor S1 to the NRS, and the NRS compares it with the available contents [26]. Afterwards, the request is forwarded to the corresponding source (router, server, and publisher), and the source sends the required content back to the user following the return path, as elucidated by steps 1 to 6.

2.3. On-Path Caching

A basic approach [27] for content placement is on-path caching, as the contents disseminate from the publisher to the consumer. In on-path caching, when a caching node receives a request for content, it responds to the consumer with a locally cached copy without any involvement of the NRS. However, this approach reduces computation and communication overhead during placement of the transmitted

content within the network [28]. If the required content does not exist, then the router forwards the request to an appropriate source, as shown by block (b) in Figure 1, from steps 1 to 6.

3. Related Studies

IoT is an auspicious environment in which smart devices are involved, where computers, watches, mobile phones, and vehicles are connected so clients can access diverse contents usually created by these above-mentioned devices [6]. The nature of data dissemination shows the most considerable transformation between the conventional Internet and IoT [29]. In the current Internet architecture, content material is shared by sending a user-generated request; meanwhile, in IoT, the content is achieved by making the system initialize an action when the scenario of interest is needed. Particularly, within the current Internet architecture, consumers generate requests for required contents at the same time that IoT gadgets generate requests by means of labeled operators [12]. The IoT packages operate well within the ICN environment, but due to the constant functionalities of IoT-based devices, present ICN caching techniques might not be implemented in IoT scenarios. The reason is that some caching techniques are too advanced for primarily IoT-based ICN scenarios [12].

3.1. Tag-Based Caching Strategy

In the tag-based caching strategy (TCS) [30], tag filters (TFs) are used for the proper matching and lookup of the requested contents to disseminate to the appropriate destination. In TCS, all the network nodes are associated with specific tag lists that help to identify highly requested contents to cache the exact location near the preferred users. The tag list is used to generate a TF by using the hash function to improve content dissemination. As the network node receives a request from the user, the corresponding TF decides whether to transmit the content cache to the intermediate location [31]. In this caching strategy, the tags associated with the desired contents are checked by the TF inside the CS, and all the tags are hashed and mapped in a counter to calculate the most requested content. If a tag counter for specific content crosses the threshold, then the content is considered popular. Consequently, all the nodes check the tags to identify the content location and decide whether the content reaches its preferred location to be locally cached. The TCS is demonstrated in Figure 2, in which user-interests are generated from sensors S1 and S2 to retrieve contents C1 and C3 from the publisher. As the predefined tag threshold is 2, the contents are therefore cached near the requested users at routers R8 and R9, because the tags' values for the requested contents cross the threshold. Thus, contents C1 and C3 will cache at routers R8 and R9 because they received several requests equal to the threshold value, as illustrated in Figure 2.

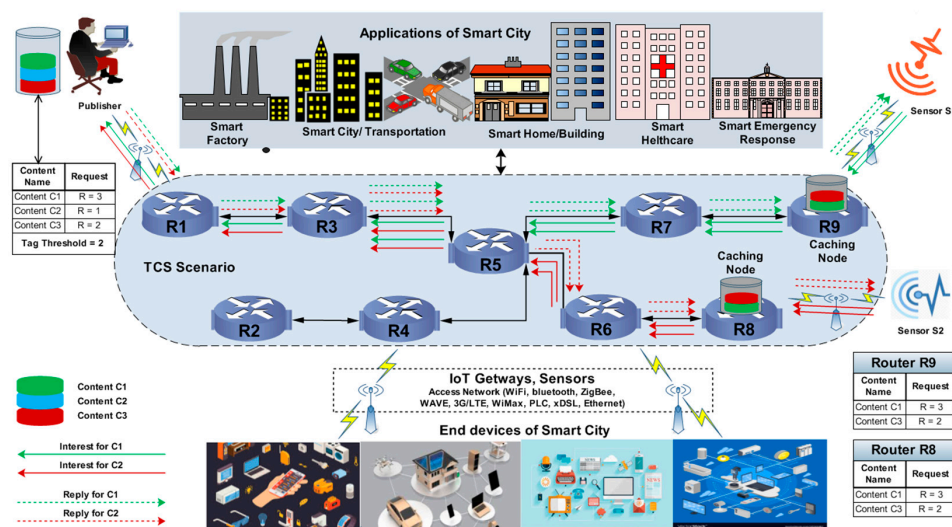


Figure 2. Tag-based Caching Strategy Scenario.

3.2. Client-Cache

The client-cache (CC) strategy [31] is known as a coherence caching strategy in which the validity of contents is considered observed for the cached contents inside the network nodes. Initially, the on-path caching approach is chosen to cache transmitted contents due to its different structure from off-path caching, and it no longer requires extra metadata to inform the client of which node is appropriate to cache the preferred content. Moreover, the number of nodes is reduced to be aware of the most essential nodes in terms of cache size and content quality. In fact, its concept is essentially derived from centrality-based caching [32], which can decrease the number of nodes and select the best nodes for caching that are associated with the maximum number of nodes in the network.

The objective of CC is to magnify the content validity, consequently called coherent caching. Content validity is examined as the requested content is found within the cache, and the content material is considered valid if its lifetime within the publisher is higher than the lifetime of its version within the cache. While selecting the requested content, a validity test checks the content inside the content publisher whenever a content material request arrives at the neighborhood-caching node. In Figure 3, multiple requests from sensors S1 and S2 are projected for contents C1 and C3. Initially, at the publisher, the lifetime of contents C1, C2, and C3 is described in tables, as represented by VC6, VC4, and VC5, respectively. According to the nature of the caching strategy, the lifetimes differ at the caching router, at 4 and 3 for C1 and C3, respectively. This is an indication to cache contents C1 and C3 because the lifetime of these contents at the publisher is greater than the lifetime of the same content at the caching router. On the other hand, content C2 has the same lifetime at both the publisher and caching router, making it invalid due to the scenario of an equal lifetime at both positions. Therefore, C1 and C3 will be cached locally at the caching router, as shown in Figure 3.

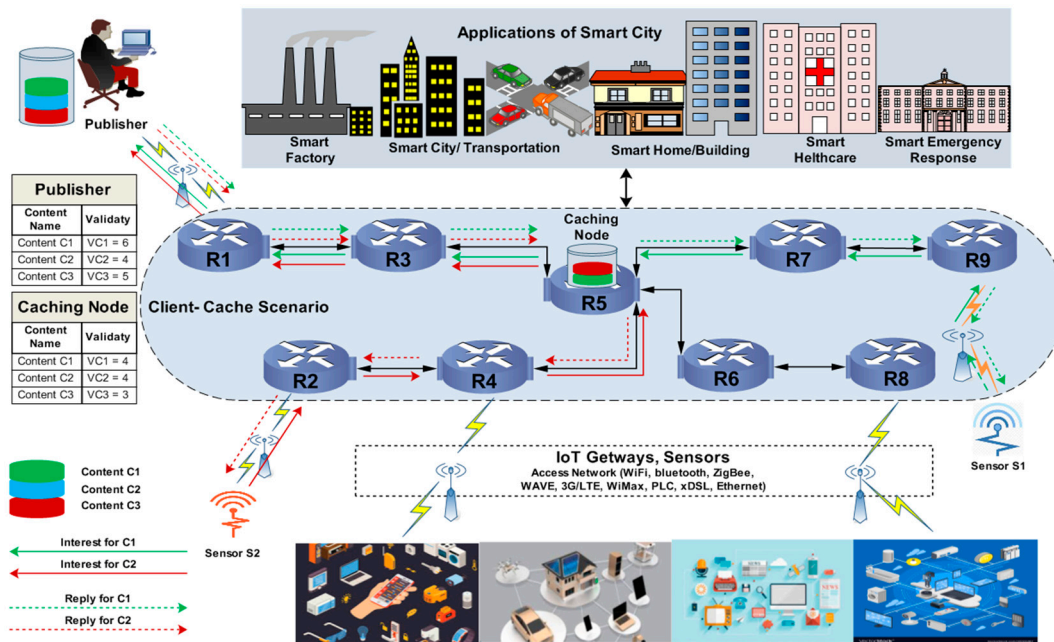


Figure 3. Client-Cache Strategy Scenario.

3.3. Problem Statement

TCS uses tags and TFs to identify the cache location for content replications. It generates homogeneous content replications by caching the same contents at multiple locations. If the publishers and subscribers are associated with the long stretch path, then the TCS increases the content retrieval latency due to limited cache locations; in turn, the cache-to-hit ratio cannot satisfy the higher demand of consumers. On the other hand, if the same content is popular again, then there is no other distinction

to stop the flooding of the same content in the network, which increases the usability of the resource (power and cache) and the bandwidth. Moreover, it demands extra cache space to manage the tags and TFs for each content at all routers, increasing the communication and searching overhead to decide which popular content will be cached at the caching router when multiple contents indicate the same popularity with limited cache size. In addition, the TCS does not care about the time to calculate the content's popularity to choose the content, which is mostly requested in a specific time interval. Furthermore, if some content is located far from the users, numerous requests for specific content are generated, but the content has low popularity at one value less than the most popular content; thus, all the requests will traverse several hops to find the requested one, which increases latency, and consequently, the stretch will maximize. Moreover, there is no other criterion for choosing popular content according to time consumption. Suppose three interests are generated for C3 in 5 s and two interests are sent for C2 in 1 s, according to the TCS, C3 will be the most popular because no time distinction is included while selecting popular content. Consequently, the most recently used content will remain unpopular, causing bandwidth congestion, which will affect the efficiency of content dissemination and boost communication delay.

Besides, all the tag lists need to be updated for each incoming interest and their size is identical in all the routers, which can cause the unpopular contents to attain a threshold for replication at multiple positions by discounting some of the interests for specific content to increase its popularity due to a limited list size. Another vigorous drawback is stretch, which is the most dominant matrix for caching; it directly affects the cache performance. In the given figure, if the interests are generated continuously for C3 and they repeatedly make it popular, it will cause the flooding of C3, and it will occupy most of the network cache. Therefore, less cache space will be vacant for new incoming contents to be cached near consumers. Hence, all the interests need to be traversed to the main publishers, and requested content is sent to the users from the main source; as a result, the stretch will increase. Due to the enormous replication of similar contents, the diversity to reduce the content retrieval delay and cache-to-hit ratio cannot retain its beneficial level in terms of strengthening the caching performance.

In the CC strategy, the aim is to improve content validity to some extent, while the caching performance in terms of stretch, content retrieval latency, and—most importantly—cache-to-hit ratio is mainly degraded. According to the strategy's algorithm, it is better for content validity to have some additional hops that increase the content retrieval latency, stretch ratio, and cache-to-hit ratio, which are the most prominent metrics to change the system performance. CC is pictorially demonstrated in Figure 3, in which two requests are projected to retrieve contents C1 and C3 from caching router R5. According to the nature of the CS, the requested content will cache at the central position, which creates significant problems due to the indentation of content to cache at only one router, such bandwidth congestion. Moreover, it increases the stretch and content retrieval latency when the cache is overflowing at the center, because all the requests need to be forwarded to the main publisher due to cache overflow. In addition, if a large number of requests are projected for content C2 and the validity of C2 is the same at both the publisher and caching router, regarding the nature of this strategy, C2 cannot replicate at the caching router. Therefore, all the requests will be forwarded to the publisher, which will maximize the stretch and latency, and the cache-to-hit ratio will be affected. If the publishers and subscribers are associated with the long stretch path, then CC increases the content retrieval latency due to a limited cache capacity at the intermediate location; in turn, the cache-to-hit ratio cannot satisfy the higher demand of consumers. Moreover, if some content is located far from the subscriber, as given in Figure 3, numerous requests for C2 are obtained from router R5, but it has low validity, at one value less than the most valid content. Thus, all the requests from R5 traverse several hops to find C2, which increases the delay and congestion and consequently heightens the overall bandwidth and stretch [7].

4. Proposed Model

In this section, the researcher proposes a flexible cache deployment strategy, i.e., the PCS, which will enhance content dissemination on the Internet and the reach of the desired information to the end users. This research proposes a cache deployment strategy to achieve maximum caching gain in terms of cache hits, and it will reduce content retrieval latency. In addition, it will minimize the distance between the source and the destination in terms of stretch. These are the popular metrics for the ICN and IoT evaluations. Moreover, the proposed strategy will contribute to an IC-IoT environment that will improve content manageability through appropriate node selection in the network. Conclusively, the proposed strategy will enhance the efficient utilization of caching.

Periodic Caching Strategy

A PCS is proposed to improve the caching performance of an IoT NDN-based caching mechanism. In a PCS, each node is associated with a distinctive statistics table in which interests are calculated to find the most frequently requested content based on content name, frequency count, recently requested time, and threshold. For the particular content name, each node counts the number of incoming interests locally. The threshold is the maximum value managed by the strategy algorithm, which is used to indicate the content most frequently requested. As the interests for specific content reach the threshold, the content is labeled the most frequently requested, and it is recommended for caching at the edge nodes of the autonomous systems (ASs). A PCS is diagrammatically presented in Figure 4. In the given figure, the most frequently requested contents are cached at the edge routers (R6 and R9). As the caches of the routers become full, some contents are evicted from the edge routers, and they follow betweenness centrality in each AS to cache at the routers associated with the highest number of interfaces during content dissemination. In this way, all the incoming interests will be forwarded through the betweenness centrality router. Moreover, if the required contents are found at the betweenness centrality router, the interests are fulfilled here rather than being forwarded to the edge routers or main publishers. For content eviction, a replacement policy named least recently used (LRU) is followed to release the cache of the awaiting content. In this way, the following goals are achieved: first, the cache-to-hit ratio is improved. Because all the interests have traversed through the betweenness central position, most of these interests will be accomplished from the central position. Moreover, the implementation of an AS provides the desired content near the consumers and decreases the path length (stretch) and content retrieval latency. In the case of memory, the heterogeneous contents are cached within an AS, and there is no replication of similar content inside the AS. Therefore, memory is used efficiently, which will reduce the content retrieval delay. In addition, the PCS reduces bandwidth congestion by decreasing homogeneous content replications. The PCS scenario is clarified in Figure 3. Two interests are generated from sensors S1 and S2 to retrieve content C1. According to the threshold, content C1 is recommended as the most frequently requested; therefore, caching at the edge routers of each AS is suggested. Additionally, the PCS presents another considerable benefit related to popular contents. Suppose contents C1 and C2 have their popularities at 700 and 800, respectively, and an incoming content (i.e., content C3) must be cached at the edge router but the cache is full. According to PCS caching, one of the cached contents has to be evicted. However, the evicted content will be cached at the betweenness centrality router; therefore, the content popularity will remain the same, which in turn minimizes bandwidth utilization.

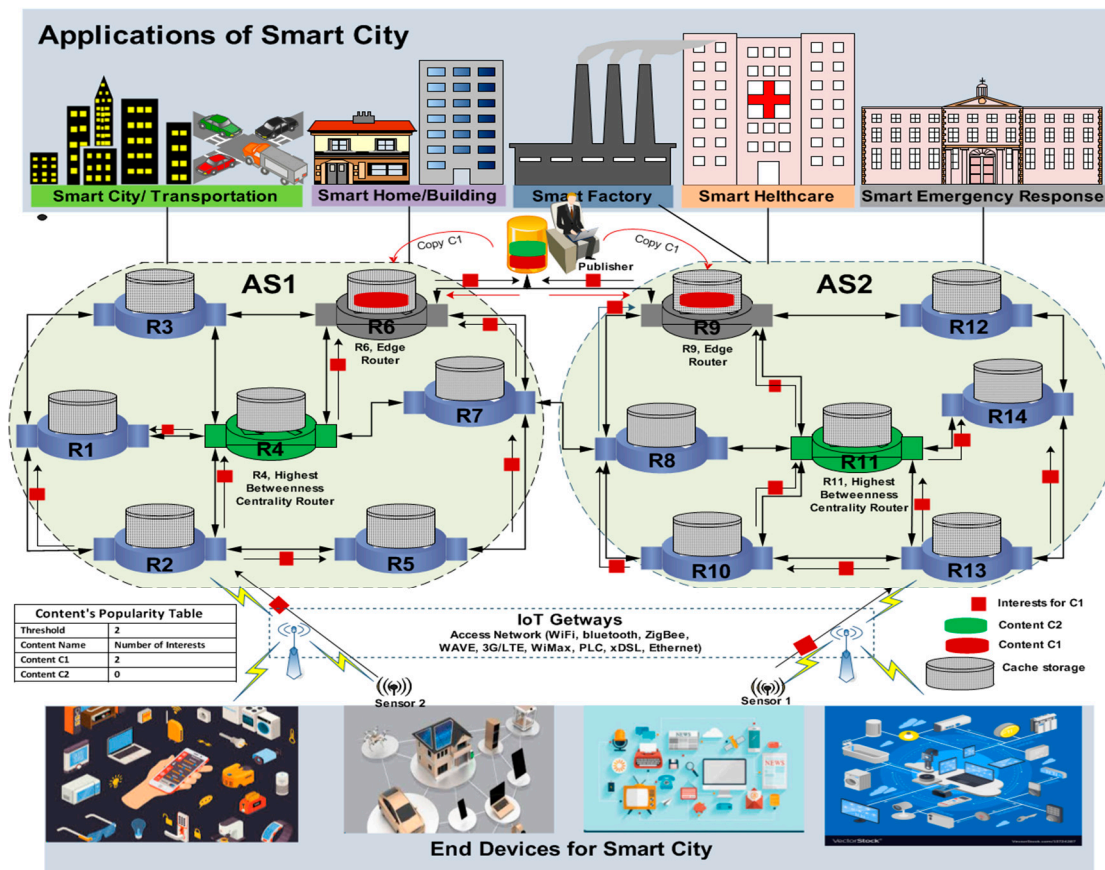


Figure 4. Periodic Caching Strategy solution for Smart City.

5. Performance Evaluation

To evaluate the proposed PCS, we created a simulation platform to evaluate the performance using the SocialCCNSim simulator [33–35], in which each node is associated with cache storage. In this paper, we use a custom-built SocialCCNSim simulator for ease of implementation, because it was built especially for NDN caching. The performance in terms of the cache-to-hit ratio, the stretch decrement, and the content retrieval latency of the proposed strategy will be compared to the strategies indicated in related works [36].

In this study, the proposed PCS is compared to well-known content popularity-based caching strategies, namely, the recently proposed CC [37] strategy, in the same simulation environment. For the simulation setup, we use the Abilene and tree topologies [38] associated with eight requesting nodes from Facebook, which uses 4039 users and 88,234 friends, in which each user counts 44 relationships. Moreover, these strategies were also checked using the tree topology. The NDN topologies differ from the k -array tree to complex topologies, or they can be a combination of both [39]. For an efficient assessment of the presented caching strategies, and to achieve fair results, an appropriate topology is needed because it has a direct impact on the simulations. We evaluate our work with different cache sizes, such as 100 and 1000 elements (1 GB and 10 GB), and 10^6 elements is selected as the catalog size. In fact, the catalog represents the total amount of elements within the network. Actually, the cache size refers to the available cache space inside each node for the temporary caching of contents at the intermediate nodes. The Zipf popularity model values of α are selected as a 0.75-associated file [40]. We select the file content because consumers have a significant tendency to access these contents, and nowadays, the dissemination of user-generated contents is huge compared to other contents [41]. The content popularity model mentions recording the popularity of individual data content. A function is used to estimate the number of interests for individual contents, and LRU [7]

was used to make room for new contents when the cache overflows. Table 1 illustrates the parameters needed to execute the simulations.

Table 1. Simulation Description Values.

Parameter	Value/Description
Simulation time	One day
Chunk Size	1 KB
Cache Size	100, 1000
Catalog Size	10 ⁶
Content Categories	File
Zipf	0.75
Topology	Abilene, Tree
Replacement Policy	LRU
Simulator	SocialCCNSim
Traffic Source	SONETOR (Facebook [42])

5.1. Cache-to-Hit Ratio

The cache-to-hit ratio presents the average quantity of existing content hits (found) as the requests are sent [43].

$$\text{CacheHitRatio} = \frac{\sum_{n=1}^n \text{Hit}_n}{\sum_{n=1}^n (\text{Hit} + \text{Miss})} \quad (1)$$

In Figure 5, graphs (a and b) elucidate the effect cache management strategies on the cache-to-hit ratio. The cache size is equally divided into 10 parts, and each part on the x-axis represents a multiple of 10 (equal to 1 GB or 2 GB). The graph depicts the cache hit for PCS increasing relatively above the rest of the other benchmark strategies. In the Abilene topology, the compared strategies show a lower cache-to-hit ratio for the cache size of 100. PCS, TCS, and CC have recorded almost the same cache hit results at 5%, 7%, and 8%, respectively, while the cache size is 100. When we expand the cache size from 100 to 1000, the PCS performs 5% and 12% better than the benchmark TCS and CC strategies, respectively, as depicted in Table 2. However, with the change in topology as a tree structure is selected, for cache sizes 100 and 1000, the cache-to-hit values reached 56%, 49%, and 45% by PCS, TCS, and CC, respectively.

Table 2. Simulation results in numeric values.

Results Metrics	Topology		Cache Size		Caching Strategies		
	Abilene	Tree	100	1000	CC	TCS	PCS
Cache-to-hit Ratio	✓		✓		0.5	0.7	0.8
	✓			✓	0.56	0.61	0.68
		✓	✓		0.6	0.6	0.8
		✓		✓	0.45	0.49	0.56
Content Retrieval Latency	✓		✓		0.44	0.42	0.36
	✓			✓	0.19	0.16	0.7
		✓	✓		0.61	0.61	0.57
		✓		✓	0.26	0.19	0.8
Stretch	✓		✓		0.48	0.46	0.42
	✓			✓	0.19	0.14	0.3
		✓	✓		0.72	0.70	0.67
		✓		✓	0.28	0.19	0.8

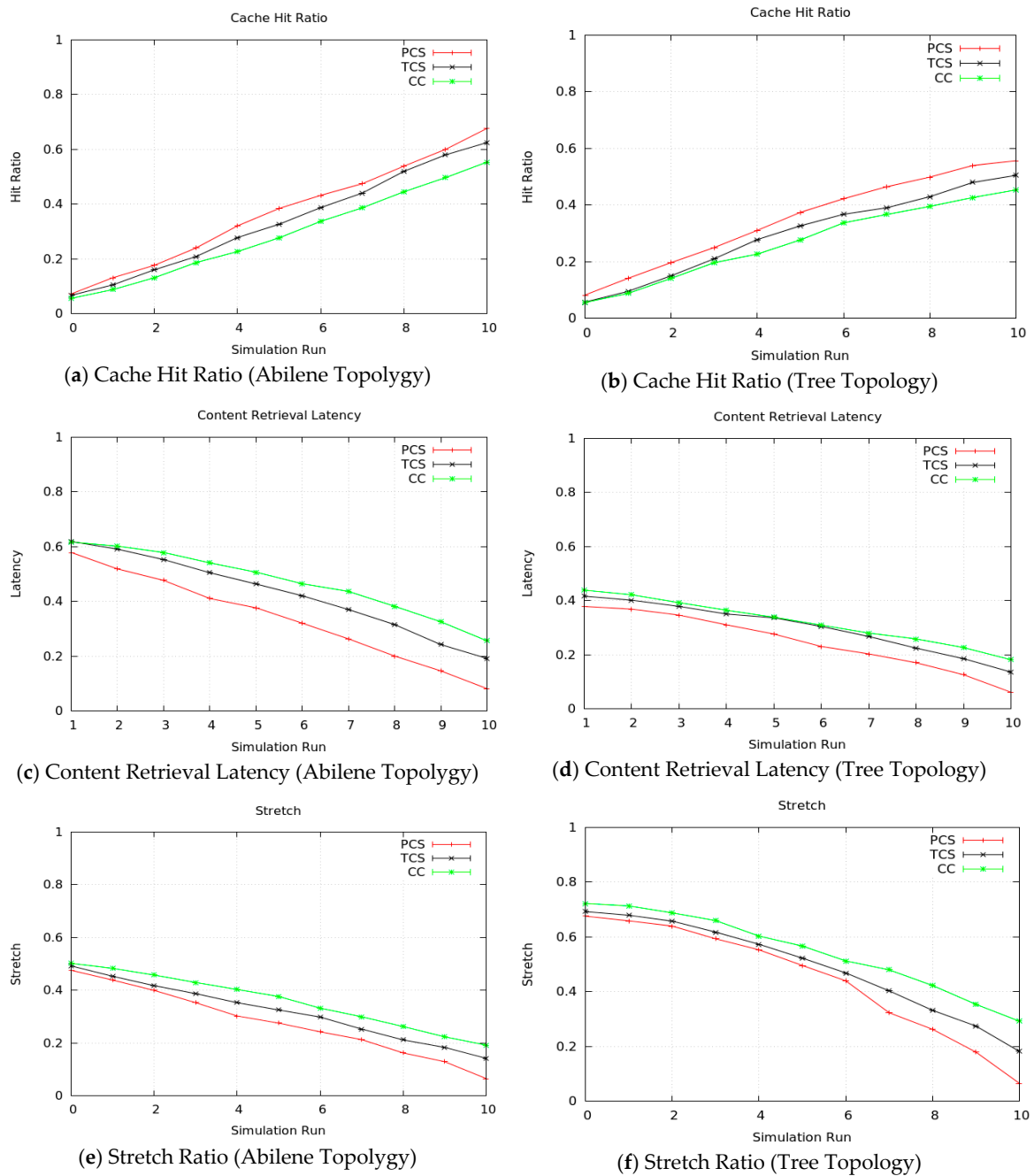


Figure 5. Simulation results caching strategies.

The PCS still performed 7% and 11% better than other strategies because of its architectural design, in which the popular contents are cached at the central position inside the AS and most incoming users' interests are satisfied there. TCS performs better than CC because it replicates the contents near the requested destination, which increases the availability of the desired contents near the users. CC shows a lower cache-to-hit ratio with respect to PCS and TCS because it selects limited nodes for the caching of valid contents, as demonstrated in Figure 3.

Moreover, TCS requires two tables (tags and TFs) to complete the task of choosing popular content, and its long duration decreases the hit rate. However, the PCS still performs well with respect to cache size in terms of achieving a better cache-to-hit ratio. Thus, we can say PCS is asymptotic to the cache size, and it can be concluded that the performance of the proposed strategy is better than that of the other benchmark strategies at hand.

5.2. Content Retrieval Latency

Content retrieval latency is the time taken by a consumer's request to reach the appropriate source and by the response to get from the source to the consumer [44]. The content retrieval time can be calculated using the below equation:

$$\text{ContentRetrievalLatency}_c = \frac{\sum_{i=1}^{|R|} \text{latency}R_i}{|R|}, \quad (2)$$

where $|R|$ is the number of requests sent by consumer c and $\text{Content Retrieval Latency}_c$ is the overall content retrieval time. Figure 5 (i.e., graphs c and d) demonstrates the effect of content retrieval latency on different caching strategies with respect to different cache sizes. When we select the cache size of 100, the performance of the proposed PCS seems 6% and 8% better than the benchmark The TCS and CC strategies have a congested data delivery path because PCS tries to cache the disseminated contents near the consumers due to the implementation of an AS. Moreover, PCS provides the best locations for caching, decreasing the distance between consumers and the desired stored contents. This shows a lower content retrieval time owing to the fewer requests that need to traverse to the main publishers in the PCS. The TCS shows higher latency results than the PCS, because it increases the amount of similar contents to be cached at multiple locations due to its nature of choosing routers near the destinations. This increases the distance when multiple requests are generated for the same content, because all the requests need to traverse to the main source, and it does not provide any central position for caching popular content. In addition, TCS needs to process two internal tables (tags table and TFs table) to decide on the intermediate nodes to cache the requested content. The duration is long; consequently, most of the incoming interests need to be forwarded to the main publisher, so the path becomes congested and the retrieval time is increased. CC performs relatively more unpleasantly than the TCS and the PCS because of its nature to cache the contents at the central position. Gradually, these contents are cached near consumers, but this increases the number of similar content replications due to increase of the interests for popular content, causing content redundancy. When we enlarge the cache size from 100 to 1000, the consequences seem changed when compared to the small cache size. However, the PCS still achieves better results, and it performed 10% and 14% better with the cache size of 1000 in terms of lessening the content retrieval time as compared to TCS and CC, respectively. However, for the tree topology, the results seem almost identical for the cache size of 100 but somehow different for the cache size of 1000, as depicted in Table 2. Therefore, we can conclude from the given outcomes, that the proposed strategy performed better than the benchmark TCS and CC strategies.

5.3. Stretch

Stretch refers to a measure of the distance from the consumer to the publisher [45]. We can calculate stretch using the below mathematical formula:

$$\text{Stretch} = \frac{\sum_{i=1}^R \text{Hop} - \text{traveled}}{\sum_{i=1}^R \text{Total} - \text{Hop}}, \quad (3)$$

where $\sum_{i=1}^R \text{Hop} - \text{traveled}$ represents the number of hops between the user and source (where the cache hit occurred), $\sum_{i=1}^R \text{Total} - \text{Hop}$ is the total number of hops (between user and server), and R is the total number of requests issued by the user. Figure 5 (graphs e and f) elucidates the stretch consequences of different caching strategies. The simulation stretch results are depicted in Table 2, which shows that the TCS and CC achieve almost the same performance (46% and 48%, respectively) at a small cache size of 100 when we select the Abilene topology. However, the PCS has a slightly low stretch value at 42%. Similarly, when we simulate these strategies with a large cache size (1000), the outcomes seem quite analogous to those of TCS and CC for the small cache size. Nevertheless, PCS improved its results to be 11% and 16% better than the TCS and CC, respectively, to achieve a

high performance in terms of reducing the distance between the source and the destination. For this reason, the PCS provides the ability to bring the desired contents closer to consumers through the selection of appropriate nodes inside the ASs. Consequently, it increases the hop decrement due to the selection of the betweenness centrality position to cache the contents at the node that is linked to the maximum number of neighbors within the ASs. The TCS and CC display a long distance because both have a high value of redundant data due to the similar content replications at multiple positions. When the cache overflows between the publisher and consumer with popular contents, a large number of interests are generated for low popularity contents that need to traverse to the main server, causing an increase in distance. As we change the topology, the tree topology is selected to be the proposed strategy because it showed an improved performance, as illustrated by the numerical results in Table 2. Therefore, the PCS expresses better results for both topologies in terms of moving the desired data near the consumers by reducing the distance between the publishers and consumers. Hence, we can conclude that the proposed strategy performs more efficiently than the benchmark strategies, as elucidated from Figure 5 (i.e., graphs e and f).

5.4. Discussion

In this study, we presented two existing caching strategies (TCS and CC) and the particular issues and challenges associated to the existing strategies in Section 3.3 (problem statement). After that, we created a new IoT ICN-based caching strategy, named the PCS, which featured a new caching architecture with respect to an AS. According to the current study, we addressed three main problems: the cache-to-hit ratio, content retrieval latency, and stretch, which are the most important metrics to improve the caching efficiency of both IoT and ICN environments. Therefore, we used a custom-built simulator named SocialCCNSim for a comparison of the proposed and benchmark strategies. The simulation results showed that the PCS performs better than the benchmark caching strategies for both cache sizes (1 GB and 10 GB) in terms of improving the cache-to-hit ratio, content retrieval latency, and stretch. According to the present study, the caching content will impact the life time of the IoT devices' batteries: a request may be satisfied by an active node while the information producer remains in its sleep mode. Due to NDN-based caching, it also addresses the security requirement and targets to secure the contents themselves rather than securing the channels connecting equipment with each other. Therefore, PCS is a promising candidate for IoT environment. It can natively support IoT scenarios while improving data dissemination and reducing total network load by decreasing the stretch between source and the users. The PCS assumes an important role in the applications of IoT environment. To this end, we have proposed in this work, a comparative study of different caching strategies, with the purpose to appoint the most suitable combination of caching strategy for IoT networks. Thus, we can say the proposed strategy performs well overall.

6. Conclusions

In this study, we focused on the advantage of IoT NDN-based caching strategies for the applications of smart city. NDN is an emerging network paradigm based on name-identified data objects and in-network caching. Therefore, NDN contents are distributed in a scalable and cost-efficient manner. With the rapid growth of IoT traffic, NDN is intended to be a suitable architecture to support IoT networks. In fact, NDN provides unique persistent naming, in-network caching and multicast communications, which reduce the data producer load and the response latency. Since the number of connected devices is increasing rapidly day by day and they heighten the requirements, this could not be accomplished by the present Internet architecture. Therefore, it is essential need to pay particular attention to the betterment of present as well as future requirements. In this case, NDN caches pay a significant contribution to solve the arising issues in an efficient way and it has become a key technology for distribution in smart city networks. The IoT NDN-based caching is accepted as an imperative part to improve the efficiency of applications included in smart city. To this end, we have proposed a new caching mechanism to disseminate data in a smart city environment to improve the

data availability procedure with less Content retrieval latency. Moreover, the proposed mechanism reduces the distance in terms of stretch and improves the cache hit ratio in efficient way. To evaluate the performance of proposed caching strategy, we compare the consequences with recently developed TCS and CC strategies. According to the given results, we can conclude that the performance of proposed strategy is much better than the benchmark strategies in hand.

Author Contributions: Conceptualization, M.A.N.; Data curation, M.A.N. and R.A.; Formal analysis, M.A.N., R.A., S.A.N. and S.H.; Funding acquisition, B.-S.K.; Methodology, M.A.N., S.A.N. and S.H.; Resources, B.-S.K.; Supervision, B.-S.K., S.A.N. and S.H.; Validation, M.A.N.; Writing—original draft, M.A.N.; Writing—review & editing, R.A. and B.-S.K.

Acknowledgments: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (No. 2018R1A2B6002399).

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

References

1. Meddeb, M.; Dhraief, A.; Belghith, A.; Monteil, T.; Drira, K. How to cache in ICN-based IoT environments? In Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 30 October–3 November 2017. [[CrossRef](#)]
2. Quevedo, J.; Corujo, D.; Aguiar, R. A case for ICN usage in IoT environments. In Proceedings of the 2014 IEEE Global Communications Conference (GLOBECOM), Austin, TX, USA, 8–12 December 2014; pp. 2770–2775.
3. Wang, L.; Afanasyev, A.; Kuntz, R.; Vuyyuru, R.; Wakikawa, R.; Zhang, L. Rapid traffic information dissemination using named data. In Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications, Hilton Head, SC, USA, 11 June 2012; pp. 7–12.
4. Braun, S.; Monti, M.; Sifalakis, M.; Tschudin, C. CCN & TCP co-existence in the future Internet: Should CCN be compatible to TCP? In Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium, 27–31 May 2013; pp. 1109–1115.
5. Arshad, S.; Azam, M.A.; Rehmani, M.H.; Loo, J. Information-Centric Networking based Caching and Naming Schemes for Internet of Things: A Survey and Future Research Directions. *arXiv*, 2017.
6. Yu, J.; Lee, N.; Pyo, C.-S.; Lee, Y.S. WISE: Web of object architecture on IoT environment for smart home and building energy management. *J. Supercomput.* **2016**, *1*–16. [[CrossRef](#)]
7. Naeem, M.A.; Nor, S.A. A survey of content placement strategies for content-centric networking. *AIP Conf. Proc.* **2016**, *1761*, 020078.
8. Yan, H.; Gao, D.; Su, W.; Foh, C.H.; Zhang, H.; Vasilakos, A.V. Caching Strategy Based on Hierarchical Cluster for Named Data Networking. *IEEE Access* **2017**, *5*, 8433–8443. [[CrossRef](#)]
9. Yu, M.; Li, R.; Liu, Y.; Li, Y. A caching strategy based on content popularity and router level for NDN. In Proceedings of the 2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC), Macau, China, 21–23 July 2017; pp. 195–198.
10. Zhang, Z.; Ma, H.; Xue, Y.; Liu, L. Fair video caching for named data networking. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
11. Xu, C.; Wang, M.; Chen, X.; Zhong, L.; Grieco, A.L. Optimal Information Centric Caching in 5G Device-to-Device Communications. *IEEE Trans. Mobile Comput.* **2018**. [[CrossRef](#)]
12. Xu, Q.; Aung, K.M.M.; Zhu, Y.; Yong, K.L. Building a large-scale object-based active storage platform for data analytics in the internet of things. *J. Supercomput.* **2016**, *72*, 2796–2814. [[CrossRef](#)]
13. Goergen, D.; Cholez, T.; François, J.; Engel, T. Security monitoring for content-centric networking. In *Data Privacy Management and Autonomous Spontaneous Security*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 274–286.
14. Qiao, X.; Nan, G.; Tan, W.; Guo, L.; Chen, J.; Quan, W.; Tu, Y. Ccnxtomcat: An extended web server for content-centric networking. *Comput. Netw.* **2014**, *75*, 276–296. [[CrossRef](#)]
15. Mohaisen, A.; Zhang, X.; Schuchard, M.; Xie, H.; Kim, Y. Protecting access privacy of cached contents in information centric networks. In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, Raleigh, NC, USA, 16–18 October 2012; pp. 173–178.

16. Dash, S.; Sahu, B.J.; Saxena, N.; Roy, A. Flooding Control in Named Data Networking. *IETE Technol. Rev.* **2018**, *35*, 266–274. [[CrossRef](#)]
17. Amadeo, M.; Campolo, C.; Molinaro, A. Multi-source data retrieval in IoT via named data networking. In Proceedings of the 1st ACM Conference on Information-Centric Networking, Paris, France, 24–26 September 2014; pp. 67–76.
18. Royon, Y.; Rangarajan, H. Temporal caching for ICN. U.S. Patents US9736263B2, 18 August 2017.
19. Wong, K.-Y. Web cache replacement policies: a pragmatic approach. *IEEE Netw.* **2006**, *20*, 28–34. [[CrossRef](#)]
20. Zhang, F.; Zhang, Y.; Reznik, A.; Liu, H.; Qian, C.; Xu, C. Providing explicit congestion control and multi-homing support for content-centric networking transport. *Comput. Commun.* **2015**, *69*, 69–78. [[CrossRef](#)]
21. Bernardini, C.; Silverston, T.; Festor, O. Socially-aware caching strategy for content centric networking. In Proceedings of the 2014 IFIP Networking Conference, Trondheim, Norway, 2–4 June 2014; pp. 1–9.
22. Ben Abraham, H.; Crowley, P. Controlling strategy retransmissions in named data networking. In Proceedings of the 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Beijing, China, 18–19 May 2017; pp. 70–81.
23. Anastasiades, C.; Schmid, T.; Weber, J.; Braun, T. Information-centric content retrieval for delay-tolerant networks. *Comput. Netw.* **2016**, *107*, 194–207. [[CrossRef](#)]
24. Ahmed, S.H.; Bouk, S.H.; Kim, D. *Content-Centric Networks: An Overview, Applications and Research Challenges*; Springer: Berlin/Heidelberg, Germany, 2016.
25. Kutscher, D.; Eum, S.; Pentikousis, K.; Psaras, I.; Corujo, D.; Saucez, D.; Schmidt, T.; Waehlich, M. *Information-Centric Networking (ICN) Research Challenges*; Internet Research Task Force (IRTF): Fremont, CA, USA, 2016.
26. Amadeo, M.; Campolo, C.; Quevedo, J.; Corujo, D.; Molinaro, A.; Iera, A.; Aguiar, R.L.; Vasilakos, A.V. Information-centric networking for the internet of things: challenges and opportunities. *IEEE Netw.* **2016**, *30*, 92–100. [[CrossRef](#)]
27. Fang, C. A survey of energy-efficient caching in information-centric networking. *IEEE Commun. Mag.* **2014**, *52*, 122–129. [[CrossRef](#)]
28. Quevedo, J.; Guimarães, C.; Ferreira, R.; Corujo, D.; Aguiar, R.L. ICN as Network Infrastructure for Multi-Sensory Devices: Local Domain Service Discovery for ICN-based IoT Environments. *Wirel. Pers. Commun.* **2017**, *95*, 7–26. [[CrossRef](#)]
29. Adhatarao, S.S.; Arumathurai, M.; Fu, X. FOGG: A Fog Computing Based Gateway to Integrate Sensor Networks to Internet. In Proceedings of the 2017 29th International Teletraffic Congress (ITC 29), Genoa, Italy, 4–8 September 2017; pp. 42–47.
30. Song, Y.; Ma, H.; Liu, L. TCCN: Tag-assisted content centric networking for Internet of Things. In Proceedings of the 2015 IEEE 16th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Boston, MA, USA, 14–17 June 2015; pp. 1–9.
31. Din, I.U.; Hassan, S.; Khan, M.K.; Guizani, M.; Ghazali, O.; Habbal, A. Caching in Information-Centric Networking: Strategies, Challenges, and Future Research Directions. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1443–1474. [[CrossRef](#)]
32. Meddeb, M.; Dhraief, A.; Belghith, A.; Monteil, T.; Drira, K. Cache coherence in machine-to-machine information centric networks. In Proceedings of the 2015 IEEE 40th Conference on Local Computer Networks (LCN), Clearwater Beach, FL, USA, 26–29 October 2015; pp. 430–433.
33. Din, I.U.; Hassan, S.; Habbal, A. SocialCCNSim: A Simulator for Caching Strategies in Information-Centric Networking. *Adv. Sci. Lett.* **2015**, *21*, 3505–3509. [[CrossRef](#)]
34. Yanqing, C.; Muqing, W.; Min, Z.; Kaili, W. Socially-aware NodeRank-based caching strategy for Content-Centric Networking. In Proceedings of the 2016 International Symposium on Wireless Communication Systems (ISWCS), Poznan, Poland, 20–23 September 2016; pp. 297–303.
35. Abdullahi, I.; Arif, S.; Hassan, S. Cache-less redundancy using hypergraph in information-centric network. *Adv. Sci. Lett.* **2015**, *21*, 3546–3549. [[CrossRef](#)]
36. Bernardini, C. SocialCCNSim. Available online: <https://github.com/mesarpe/socialccnsim> (accessed on 23 March 2017).
37. Ud Din, I. Flexpop: A popularity-based caching strategy for multimedia applications in information-centric networking. Ph.D. Thesis, Universiti Utara Malaysia, Sintok, Malaysia, 2016.

38. Ibrahim, A. Proxcache: A New Cache Deployment Strategy in Information-Centric Network for Mitigating Path and Content Redundancy. Ph.D. Thesis, Universiti Utara Malaysia, Sintok, Malaysia, 2016.
39. Bernardini, C.; Silverston, T.; Festor, O. A comparison of caching strategies for content centric networking. In Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USA, 6–10 December 2015; pp. 1–6.
40. Rossi, D.; Rossini, G. Caching performance of content centric networks under multi-path routing (and more). *Relatório Técnico Telecom ParisTech* **2011**, 1–6.
41. Evans, D. *Social Media Marketing: An Hour a Day*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
42. Cantador, I.; Brusilovsky, P.L.; Kuflik, T. *Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011)*; ACM: New York, NY, USA, 2011.
43. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A survey of information-centric networking research. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1024–1049. [[CrossRef](#)]
44. Abdalla, B.M.A.; Hamdan, M.; Mohammed, M.S.; Bassi, J.S.; Ismail, I.; Marsono, M.N. Impact of Packet Inter-arrival Time Features for Online peer-to-peer (P2P) classification. *Int. J. Electr. Comput. Eng. (IJECE)* **2018**, *8*, 2521–2530.
45. Meddeb, M.; Dhraief, A.; Belghith, A.; Monteil, T.; Drira, K.; AlAhmadi, S. Cache Freshness in Named Data Networking for the Internet of Things. *Comput. J.* **2018**. [[CrossRef](#)]



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).