applied sciences

MDPI

Article

An Incrementally Deployable IP-Compatible-Information-Centric Networking Hierarchical Cache System

Li Zeng [1,2], Hong Ni [1,2] and Rui Han [1,2,*]

1 National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, No. 21, North Fourth Ring Road, Haidian District, Beijing 100190, China; zengl@dsp.ac.cn (L.Z.); nihong@mail.ioa.ac.cn (H.N.)

2 School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, China

* Correspondence: hanr@dsp.ac.cn; Tel.: +86-138-1107-7380

Received: 10 July 2020; Accepted: 3 September 2020; Published: 8 September 2020

Abstract: The major advantage of information-centric networking (ICN) lies in in-network caching. Ubiquitous cache nodes reduce the user's download latency of content and the drain of network bandwidth, which enables efficient content distribution. Due to the huge cost of updating an entire network infrastructure, it is realistic for ICN to be integrated into an IP network, which poses new challenges to design a cache system and corresponding content router. In this paper, we firstly observed that the behavior pattern of data requests based on a name resolution system (NRS) makes an ICN cache system implicitly form a hierarchical and nested structure. We propose a complete design and an analytical model to characterize an uncooperative hierarchical ICN caching system compatible with IP. Secondly, to facilitate the incremental deployment of an ICN cache system in an IP network, we designed and implemented a cache-supported router with multi-terabyte cache capabilities. Finally, the simulation and measurement results show the accuracy of proposed analytical model, the significant gains on hit ratio, and the access latency of the hierarchical ICN cache system compared with a flat cache system based on naming routing, as well as the high performance of the implemented ICN router.

Keywords: ICN; in-network caching; hierarchical cache system; router

1. Introduction

Due to the advantages of multicasting, mobility, security, and caching, information-centric networking (ICN) is considered an effective way to improve the existing network from network architecture level [1]. Compared with a traditional IP network, ICN cares more about content itself than the location of content. Following this principle, ICN decouples content addressing from routing by separating the identifier (name) and the locator of content. Nevertheless, many researchers believe that the benefits advocated by ICN require magnificent upgrades to entire network infrastructures [2], the huge costs of which make it impractical for clean-state ICN to be deployed.

In order to smoothly evolve, many studies have shown interest in incremental deployment of ICN (and correlative cache system) without changing today's IP infrastructure as much as possible [2–7]. Some ICN paradigms are implemented as an overlay top of the current internet, and these effectively utilize existing IP facilities and routing schemes, such as MobilityFirst (MF) [3] and the network of information (NetInf) [4]. The former focuses on supporting the seamless handover of terminals in mobility scenarios, and the latter was funded by the 4WARD project and aims to improve large-scale content distribution. Meanwhile, an ICN router should be able to forward and process both IP and

ICN flows at an Layer 3 (L3) network layer [5]. Another aspect of the deployment scheme is that of ICN over IP [6,7] based on the gateway isolation, where the first link from the user's device to network uses existing IP-based protocols, such as HyperText Transfer Protocol (HTTP), Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). As an entry/out point of ICN, the gateway converts chosen protocol to ICN. However, the improvement of ICN over IP depends on the limited scope of the gateway, which makes it difficult to implement new services and applications in an entire network.

There are two typical content discovery mechanisms in ICN. One is routing-by-name, which couples name resolution with content routing in a data plane. For instance, the content-centric network (CCN) [8,9], the most widely studied ICN paradigm whose routers forward user's requests to the right port computed by content name, adopts a routing-by-name mechanism. The shortcomings of this mechanism are as follows. First of all, routers lack the capacity to save routing information for considerable amount of content. In addition, it is easy to cause large-scale routing updating when the state of each duplicate changes. The second content discovery mechanism is lookup-by-name, which accomplishes content discovery by retrieving the network addresses (NAs) of the named data object (NDO) from a dedicated device named the name resolution system (NRS). The lookup-by-name mechanism makes routers concentrate on routing and caching at the cost of signaling overhead from the NRS, which alleviates the computing pressure of routers.

With the assist of resolution nodes (RNs) distributed around the network, deploying an NRS is an effective method to encourage ICN to coexist with the current IP network with high scalability [10]. A hierarchical NRS reduces the inter domain traffic and the latency of naming resolution by keeping the name information of content as close to the user as possible. Communication between the RNs of different levels can be realized by deploying a multi-level distributed hash table (MDHT) [11] in each RN. For example, the data-oriented network architecture (DONA) [12] deploys more than one logical resolution handler (RH) in every autonomous system (AS). In a hierarchical NRS, we have observed that the request packet of content name resolution is sent to a higher-level RN if the request cannot be satisfied in a lower-level RN. In other words, in this behavior pattern, the request of the NDO is sent to a higher-level cache node if it cannot be hit in a lower-level one. Hence, the mechanism of content addressing based on a hierarchical NRS makes an uncooperative ICN cache system implicitly form a hierarchical structure.

As another crucial component of a caching system, an ICN router is supposed to be designed with multiple factors including IP compatibility, a communication interface with an NRS, a transparent caching function based on the chunk level rather than the packet level [13], and an ICN transport protocol stack to ensure reliable service for users. However, few studies have comprehensively considered these aspects when designing ICN routers.

In summary, the contribution of our work is as follows:

- We designed a simple uncooperative IP-compatible-ICN cache system based on an NRS that implicitly forms a hierarchical and nested structure because of the behavior pattern of data requests. Furthermore, we propose a modeling approach to analysis this hierarchical cache system, which considers the correlation of arriving requests between adjacent cache nodes under different caching strategies.
- We designed a cache-supported ICN router with a spilt architecture as a complete service unit in our cache system, which comprehensively takes both forwarding and caching service into consideration. Furthermore, we proposed an implementation scheme of a high performance ICN router with multi-terabyte cache capabilities in a multi-core environment of a general ×86 server.
- We conducted extensive experiments to verify the accuracy of our analysis model for the cache hit ratio at different levels and performed a thorough evaluation of hierarchical cache system (HCS) performance. Experimental results demonstrated the significant gains of the hit ratio and access latency of HCS in comparison to other kinds of cache systems under different caching strategies.

The rest of this paper is organized as follows. Section 2 provides related research to our work. The design and model analysis of the HCS is presented in Section 3. Section 4 presents the design and implementation details of cache-supported router. Our experimental results are shown in Section 5. Finally, we conclude the paper and describe our future work in Section 6.

2. Related Work

2.1. Cache System

The research of ICN cache systems has mostly focused on the design and implementation of caching strategy, such as leave copy down (LCD) [14], leave copy everywhere (LCE), leave copy with probability (LCP) [15], cache less for more (CL4M) [16], which are lightweight, uncooperative strategies with low computational complexity. There are also some strategies that are implemented with a high memory overhead. For instance, Saino L. proposed hash caching [17], which makes cache decisions according to the caching state of content itself and the follow-up cache nodes. For popularity-based caching strategies, Naeem M. A. proposed a time-based mathematical function to select the caching content, which is the most frequently requested content in the given time interval [18]. For centrality-based caching strategies, Meng Y. proposed an efficient hybrid strategy to reduce multiple replications of the same content at the data dissemination path according to the judgement of an improved betweenness centrality of router [19].

The modeling technology of traditional cache systems and related theoretical research has been widely studied. Che et al. [20] proposed a modeling method to deduce the cache performance of different level cache nodes based on individual document requests following arbitrary access frequency distribution, and they found out the fundamental design principles for hierarchical web caching. Lev B et al. [21] presented an optimized model of a hierarchical cache system for unbalanced traffic in a content delivery network (CDN). The aim of his model is to deduce how much memory must be allocated for cache node to achieve maximum cost-effectiveness.

Compared with traditional caching systems (such as Web, CDN, Peer to Peer (P2P)) that only support a single traffic type, an ICN caching system has new features. Specifically, ICN supports a transparent and universal caching service for multiple upper-layer applications. The basic unit of ICN caching is no longer a file or a variable segment of a file; instead, it is a smaller data chunk with a global unique content identifier. As a counterexample, although the web cache system adopts the open HTTP protocol, the same object cannot be consistently identified between different domains.

Due to the change of traffic characteristics from the object-level to the chunk level, it is difficult to directly apply the modeling and analysis method of a traditional cache system to ICN. Only a few studies have extended the modeling method of an ICN cache system. Christine et al. [22] proved that the Che approximate approach we introduced above is suitable for estimating the performance of a single cache node in ICN through mathematical demonstration. Taking the effects of temporal locality into account, Martina et al. [23] extended the Che approximate to model any cache node in a network with general (mesh) topology, but they only proposed a unified methodology under the LCE strategy. Additionally, Psaras et al. [24] developed a mathematical model for a single router based on continuous time Markov-chains that provided a new idea for modeling ICN.

2.2. Cache-Supported ICN Router

Improving the performance of forwarding and caching are two major research directions for ICN routers. Perino et al. [25] firstly designed a content router under the routing-by-name mechanism based on the longest-prefix matching algorithm, and then they implemented a high-speed forwarding module in a multi-core environment. ICN is strict with routers in forwarding capacity for line speed, but a slow-speed block Input/Output (I/O) operation in a forward path would be a burden. To eliminate the adverse effects of the forwarding process brought by the block device, Ding firstly proposed a split architecture [26] that decouples the router into two independent processes: storage end and switch end. Moreover, Ding designed a switch end and a storage end communication protocol (SSCP) to solve the packet dependency problem caused by the split architecture. However, the protocol conversion between the SSCP and IP also brings performance problems.

Furthermore, researchers have hoped to extend cache size to Terabytes (TBs) through SSDs (solid state disks) or HDDs (hard disk drives). L. Saino firstly proposed the complete design of a hierarchical cache module with Dynamic Random Access Memory (DRAM) or SSD and conducted a detailed performance evaluation [27], but he did not take the process of forwarding into account. Rossini argued that the bottleneck of the hierarchical cache architecture is the high access latency of the block device [28]. Therefore, according to the characteristics of video traffic, he proposed a proactive prefetch mechanism that prefetched subsequent chunks of recently requested chunks from the L2 (SSD) to the L1 cache (DRAM) that took advantage of the correlation between chunks based on hierarchical naming. Despite the prefetch mechanism transforming the bottleneck from the access latency of the SSD to external data rate, we believe it has obvious defects because it ignores the existence of considerably long tail loads in the network. Frequent invalid prefetch operations occupy a large amount of memory space. If the size of each I/O operation is less than a page, which is the basic unit of read and write operations in SSDs, the utilized I/O bandwidth will suffer. The authors of [29] leveraged the Linux vector I/O to gather chunks scattered across threads, and then they combined these chunks into a segment whose size was equal to one page. This solved the problem caused by I/O operations with small size. Additionally, since the end property has been attached to ICN router in a new approach, many recent studies have tried to solve the problems caused by congestion. A typical example can be found in [30], where Zafar H. proposed a fairness-based active queue management algorithm to ensure the fair allocation of resources by interest rate shaping.

3. System Design

In this chapter, we firstly outline the design of a hierarchical ICN caching system that is compatible with IP. Secondly, we detail a complete caching mechanism based on a hierarchical NRS. Finally, we model the hierarchical cache system and analyze the hit probability of each level under different cache strategies.

3.1. System Overview

Figure 1 shows a simple hierarchical cache system based on a hierarchical NRS with 3 layers. There are two crucial components in a cache system—NRS and cache-supported router. An NRS provides a basic service of name resolution and divides a network into plenty of hierarchical nested areas (HEAs) where at least one dedicated RN is deployed. Content sources (including cache-supported routers and content publisher) in network register the mapping of its NA and content name in the NRS. When users request the NDO, it firstly asks RN that belongs to the lowest HEA for the NA where the content located. The query does not stop forwarding to the upper level RN until it is satisfied in some RN. This name resolution process can be seen as a request of the NDO to travel from routers in given lower level HEAs towards routers in the upper level HEAs until the requested NDO is found, which makes an ICN cache system implicitly form a hierarchical structure. To make an ICN cache system compatible with IP, we implemented ICN by overlaying it onto IP and identifying IP (IPv4 or IPv6) addresses as content locators, i.e., NA. A 160-bit flat entity ID (EID) was employed as the content name due to its characteristic of self-verification. ICN routers need to be able to process both IP and ICN flows (including routing and caching the NDO), as well as to make caching decisions locally. Due to the substantial overhead brought from small-sized chunks [31], we set the size of the chunk transmitted in network to up to a few MBs.

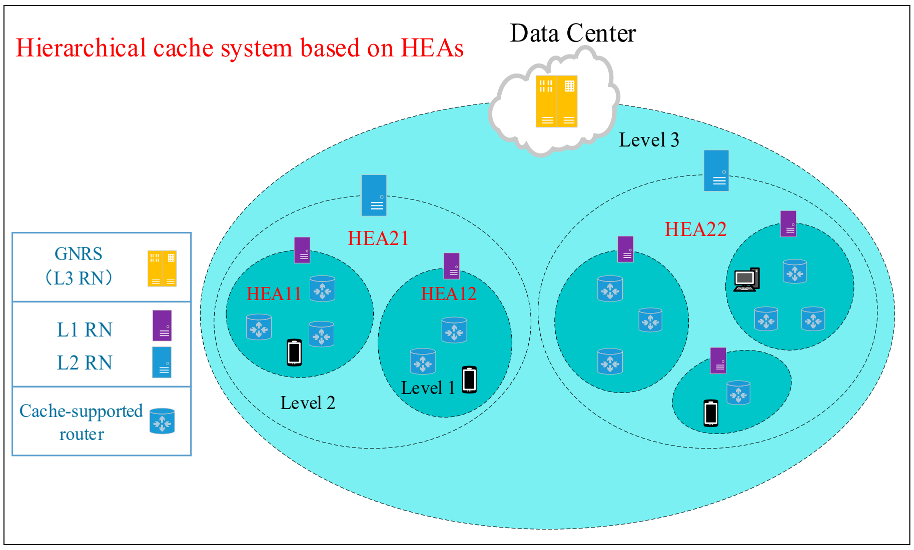

Figure 1. Overview of a hierarchical cache system.

3.2. *Cache Mechanism Based Hierarchical NRS*

Then, to showed how the hierarchical IP-compatible-ICN cache system works, we designed a specific caching mechanism (As shown in Figure 2) based on an enhanced name resolution system (ENRS) [10,32] with a hierarchical structure, mainly to provide low-latency resolution services for certain delay-sensitive applications, such as 5th generation mobile networks (5G) or mobility support [33]. An ENRS divides an HEA into finer grains by quantifying the transmission latency constraints from the user to the resolution node, which means that the same level HEAs have the same resolution latency and the higher the level of HEAs, the longer the latency is.

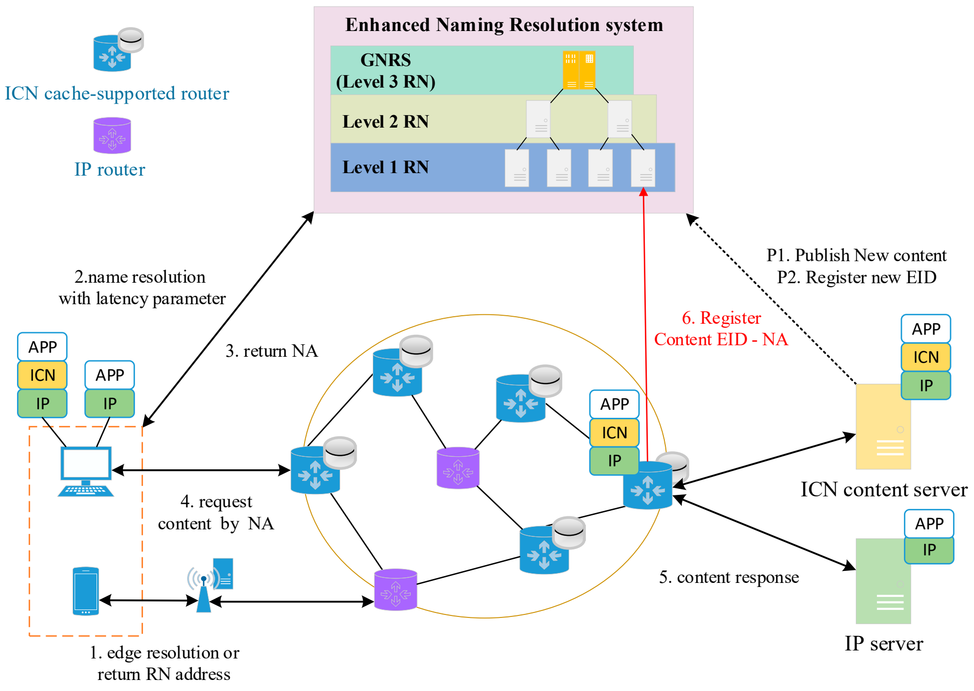

Figure 2. A specific caching mechanism based on an enhanced name resolution system (ENRS).

To advertise new content, content providers register the EID of the content to a global name resolution service (GNRS) so that users in the network can discover it (steps P1 and P2). A GNRS provides a name resolution service for all network elements in a whole network, which can be regarded as the Level 3 RN. To request content, like with mobile terminals, they must firstly attempt to resolve the content name (EID) in the ENRS agent deployed in the point of attachment (POA) of the edge of the internet. To further reduce the resolution latency, the ENRS agent saves the mapping of some popular content EID. If the lookup fails, the NAs of the nearest RN of each level are returned (step 1). As for fixed network users, the Dynamic Host Configuration Protocol (DHCP) is a valuable method to automatically configure the NAs of RHs.

Afterwards, the user resolves the content name with the latency parameter [10] according to the application requirements to the RN (step 2). If there is no needed name information in the RN corresponding to the latency parameter, the request of resolution is resent to an upper level RN. Multiple NAs where the content is cached in this HEA are returned to the user when the resolution is successfully completed (step 3). Subsequently, the user sends a request to an optimal NA selected by a source selection mechanism (SSM) (step 4). After receiving the request, the content source splits the requested NDO into ICN packets encapsulated with IP headers according to the network maximum transmission unit (MTU), and then it transmits them on the network (step 5). Along the chunk transmission path, ICN routers collect and assemble ICN packets into a complete chunk. If chunks have been assembled completely and the EID has been successfully verified, ICN routers decide to cache it and register the mapping of the chunk EID and the NA of router to the RNs of the located HEAs of every level (step 6).

Source Selection Mechanism

We propose two simple designs of SSMs. The first approach to select preferable content source is to select the nearest copy. This method firstly judges the distance of each source by a traditional topology-based routing scheme based on shortest path algorithms. However, selecting the nearest source raises communication costs, which increase at an extremely high rate with the growth of the number of the alternative sources.

The second alternative to investigate is the random selection of the source, which is a simple method to save the latency of judging distance between nodes. Due to strict requirements of the algorithm that divides the network into HEAs with different levels in the ENRS, there cannot be many cache nodes located in the lower HEA that is closest to user. Hence, we inferred that random selection may have a similar performance to that of nearest selection.

3.3. System Model

Considering a three-layer hierarchical cache system shown in Figure 1, we attempted to model it and analyze the cache-hit probability of the HEAs at all levels. Che approximation, which our model builds upon, is a simple and efficient method to estimate the hit probability of a cache node using the least recently used (LRU) replacement strategy [22]. Firstly, we briefly introduce the Che approximation.

3.3.1. The Che Approximation

Suppose that there is a cache node k in a network with cache capacity M (count by chunks). Users request NDO c from a set C $(1 < c < C)$ with a large, fixed-size catalog. The requests of c arrive k with rate λ_c according to a homogeneous Poisson process. The request sequence follows the Independent Reference Model (IRM), which means the request probability for c is fixed and is independent of past requests. In other word, the popularity of content with a Zipf-like law does not change with time. In the real world, it seems unreasonable that there is no correlation between the request probability and the content itself. However, the correlation can be ignored when the content catalog C is large enough [22].

The time average cache-hit probability that the c can be found in k can be expressed as:

$$P_{hit}(c) = P_{in}(c) = 1 - e^{-\lambda_c T_M} \tag{1}$$

where T_M, which is a constant independent of c, is a characteristic time that receiving M different requests (excluding c) in $k.T_M$ represents the living time duration for c in k. The average hit probability of cache node k is:

$$P^k_{hit} = \sum_C \frac{\lambda_c}{\Lambda} P_{hit}(c) \tag{2}$$

where $\Lambda = \sum_C \lambda_c$ is the total arrival rate of all content requests. In the above equation, the only unknown value is characteristic time T_M, which can be easily obtained with arbitrary precision by a fixed point procedure.

3.3.2. Optimal Che Approximation

The Che approximation only considers a simple situation of a single cache node while it ignores the correlation of requests between adjacent nodes. The key to applying Che approximation to a general network node i is to analyze all possible events that happened in the duration of $\left[t - T^i_M, t\right]$ for NDO c in node i at time t [23]. The correlation of requests between adjacent nodes differs under different cache strategies that affect possible events that happened in a different way. We optimized the Che approximation by analyzing LCP, LCE, and LCD one by one, as these are the most widely used on-path caching strategies in ICN.

Suppose the cache node j belongs to set J and is adjacent to node i. This implies that j is located behind i in a content transmission path so that j forwards the miss request flow of c to i. Our inference was based on a hypothesis: $T^i_M > T^j_M$, otherwise $P^i_{hit} = 0$ because the request of c will hit the cache node j. Therefore, the average arrival rate of requests for c at cache node i is equal to:

$$\overline{\lambda}_c(i) = \sum_{j \in J} \overline{\lambda}_c(j)\left(1 - P^j_{hit}(c)\right) \tag{3}$$

We first analyzed the optimal Che approximation in LCP, because LCE is only a special case of LCP when $p = 1$.

Leave-Copy-Probability

As for LCP, each node along a transmission path caches content with a fixed probability p, which can be adjusted according to the cache condition in network. Suppose that a request of c hits node i at time t, which means that NDO c is stored in i rather than j.

Hence, the previous request of c arrives i at time duration $\left[t - T^i_M, t\right]$, and this may bring about two possibilities: One is that node i has a hit, and the other is that node j caches c with probability p when the request missed in i. We estimated time average probability $P^i_{in}(c)$ by the standard Che approximation of request rate $\overline{\lambda}_c(i)$ without considering the variation of T^i_M because the T^i_M for single node is hard to change when the content catalog C is large enough.

We obtained:

$$P^i_{in}(c) \approx \left[P^i_{hit}(c) + p\left(1 - P^i_{hit}(c)\right)\right]\left(1 - e^{-\overline{\lambda}_c(i)T^i_M}\right) \tag{4}$$

$P^i_{hit}(c)$ cannot be simply estimate like $P^i_{in}(c)$ because it is changeable and is influenced by the correlation between nodes. Additionally, to ensure that there is no copy of c cached in node j at time t, there are 3 possible events that could have happened in the duration of $\left[t - T^i_M, t\right]$. The last request of c either arrives i from given j at the time duration of $\left[t - T^i_M, t - T^j_M\right]$, it arrives at $\left[t - T^j_M, t\right]$ while failing to trigger cache action for c with probability $1 - p$, or it arrives i from any node $k(k \in J)$ different from j at $\left[t - T^i_M, t - T^j_M\right]$.

The latter two possible events can be regarded as standard Poisson flow, but the first is not because it depends on the cache state of NDO c at node j. Thus, we write the hit conditional probability of node i from the given node j as:

$$P^i_{hit}(c|j) = \left[P^i_{hit}(c) + p\left(1 - P^i_{hit}(c)\right)\right]\left(1 - e^{-A_{i,j}}\right) \tag{5}$$

where

$$A_{i,j} = \overline{\lambda}_c(j)\left(1 - P^j_{in}(c)\right)\left(T^i_M - T^j_M\right) + (1-p)\overline{\lambda}_c(j)\left(1 - P^j_{in}(c)\right)T^i_M + \sum_{k\in J}^{k\neq j}\overline{\lambda}_c(k)\left(1 - P^k_{in}(c)\right)T^i_M$$

Note that Equation (5) can be simplified to the following equation for deducing LCE when $p = 1$:

$$P^i_{hit}(c|j) = \left(1 - e^{-A_{i,j}}\right) \tag{6}$$

where

$$A_{i,j} = \overline{\lambda}_c(j)\left(1 - P^j_{in}(c)\right)\left(T^i_M - T^j_M\right) + \sum_{k\in J}^{k\neq j}\overline{\lambda}_c(k)\left(1 - P^k_{in}(c)\right)T^i_M$$

Equation (6), which we deduced for LCE, was same as the conclusion from [23]. The final result of $P^i_{hit}(c)$ could be obtained by de-conditioning the form iteration of each node j in J.

Leave-Copy-Down

For LCD, the content is cached only in the next node of the hit node for each time, which avoids the existence of multiple copies of the same content in the transmission path. Assume that $P^i_{in}(c) \approx \left(1 - e^{-\overline{\lambda}_c(i)T^i_M}\right)$. There is a known strong assumption that NDO c can be inserted in j only if c has cached in i already. Hence, the two possible conditions to make NDC c cache in node j at time t are the previous request either hit at i forwarded from j or hit in the j at the time duration of $\left[t - T^j_M, t\right]$. We deduce:

$$P^J_{in}(c) \approx \left[\left(1 - P^J_{in}(c)\right)P^i_{hit} + P^J_{in}(c)\right]\left(1 - e^{-\overline{\lambda}_c(j)T^j_M}\right)$$

Moreover, to ensure that the request of c forwards to i at time t, there are 3 possible events that could have happened in the duration of $\left[t - T^i_M, t\right]$. The last request of c either arrives i from given j at the time duration of $\left[t - T^i_M, t - T^j_M\right]$ (provided that $T^i_M > T^j_M$) when NDO c is known not to be cached in node j before, it arrives i at $\left[t - T^j_M, t\right]$ because c stores in neither i nor in j, or it arrives i from any node $k (k \in J)$ different from j. Thus, we write:

$$P^i_{hit}(c|j) = \left(1 - e^{-\overline{\lambda}_c(i)(1-P^J_{in}(c))(T^i_M - T^j_M)}\right)e^{-\overline{\lambda}_c(i)T^j_M} + \left(1 - e^{-A_{i,j}}\right) \tag{7}$$

where

$$A_{i,j} = \overline{\lambda}_c(j)\left(1 - P^j_{in}(c)\right)T^i_M + \sum_{k\in J}^{k\neq j}\overline{\lambda}_c(k)\left(1 - P^k_{in}(c)\right)T^i_M$$

We needed to combine de-conditioning and a multi-variable fixed-point approach [34] to solve the entire conditional probability equation.

3.4. Hit Probability of HEAs in Hierarchical Cache System

We attempted to apply the optimal Che approximation to analyze the hit probability of HEAs in the three layer cache system depicted in Figure 1. We mainly analyzed the L1 and L2 HEAs, while the hit ratio in the L3 HEA was equal to 1. We assumed that the set $J(j_1, j_2, \cdots, j_n)$ represents all the

L2 HEAs in the network and there was an L1 HEA $j_{kk}(j_{kk} \in J_k(j_{k1}, j_{k2}, \cdots, j_{kn}))$ located in L2 HEA J_k. Hence, $P^{L1}_{hit}(m, j_{kk})$ is the hit probability of L1 HEA j_{kk}, which can be written rapidly as:

$$P^{L1}_{hit}(j_{kk}) = \sum_{i \in I_{kk}} \sum_{C} \frac{\lambda_c(i)}{\Lambda} P^{i}_{hit}(c) \tag{8}$$

At the same time, the hit probability of L2 HEA J_k is:

$$P^{L2}_{hit}(m, J_k) = \sum_{b \in J_k}^{b \neq j_{kk}} \sum_{i \in I_{kk}} \sum_{C} \frac{\lambda_c(i)}{\Lambda} P^{i}_{hit}(c) \tag{9}$$

The value of $P^{i}_{hit}(c)$ is related to the different caching strategies and source selection mechanisms we discussed earlier. There is no correlation of requests between adjacent nodes when the user randomly selects a source. This implies that the hit rate can be estimated by common Che approximation because the request flows arriving to any cache node in each HEA can be modeled as a standard Poisson flow. Hence, we easily replaced the $P^{i}_{hit}(c)$ with Equation (1). As for the mechanism of selecting the nearest source, the optimal Che approximation can work because requests between cache nodes are relevant. Therefore, we replaced the $P^{i}_{hit}(c)$ with results deduced from Equations (5) and (7) under different caching strategies.

4. Cache-Supported ICN Router

In this part, we firstly propose the design of a cache-supported ICN router that is compatible with IP and works in our cache system. After that, we provide complete high-performance implementation details.

4.1. Architecture Design

Software-defined networking (SDN) [35] is widely used in designing ICN routers because it is able to support new network protocols without upgrading the network equipment. Recent research has proposed some new SDN technologies, such as protocol oblivious forwarding (POF) [36] and programming protocol-independent packet processors (P4) [37]. POF technology adopts the matching method of <offset, length>, which can support almost any new protocol including those customized by developers themselves. Hence, we selected POF to develop the forwarding function in our router.

Figure 3 depicts the software architecture comprised by various function modules such as *packet routing*, *cache service*, *cache management*, and *ICN transport protocol stack*.

The *cache service* module consists of two submodules, chunk assemble and request service, which are the two aspects of read and write of chunk operation.

***Chunk assemble* submodule**: Due to the limitations of network MTUs, a chunk is segmented into packets of different sizes for transmission. The problem then is that incoming packets belonging to same chunk may be lost or out of order when routers collect ICN packets in a complex real network. Therefore, the *chunk assemble* submodule assembles packets with same name into a complete chunk, and then it verifies the chunk's integrity and the validity of the name. The process of chunk assembling brings inevitable delays, which makes it impossible to complete the cache operation while guaranteeing wire-speed forwarding. We adopted a split structure [26] to decouple low-speed block device I/O operations from the forwarding path. Consequently, a cache-supported router is decomposed into two separate processes: a forwarding element (FE) and a cache element (CE), between which both messages and chunks pass via inter-process communication.

***Request service* submodule**: When a router receives a data request packet, the *request service* submodule fetches the requested chunk from the disk by sending a message to the CE. The chunk is put into an ICN transmission protocol stack for specific sending.

***Packet routing* module**: The *packet routing* module in an FE includes two submodules: The *IP routing* submodule and the *ICN routing* submodule, which forward the ICN and IP flows according to the flow rules issued by the controller. In addition, the *ICN routing* submodule makes caching decisions for passing packets based on certain caching strategy.

ICN transport protocol stack: The *ICN transport protocol stack* provides reliable and low latency transmission based on chunks. Appropriate transport protocols containing retransmission and congestion control guarantee the quality of service (QOS) of ICN. For example, the router shapes the interest rate when it detects occurring congestion via an active queue management algorithm [30]. We do not expand on the relevant design details since they are not the focus of this paper.

Cache management module: The *cache management* module mainly manages chunks cached in the CE, which includes three submodules. The *cache replacement* submodule replaces the unpopular chunks following the LRU policies when the disk space is insufficient. The *cache index* submodule is a hash table used to store the mapping of content name and store address. To speed up the lookup, we set the size of each bucket in hash table to the size of the Central Processing Unit (CPU) cache line. The *register/logout* module is responsible for communicating with the NRS. When a chunk is successfully written into disk or deleted, this submodule needs to register/logout the mapping of the content EID and NA of router in the NRS.

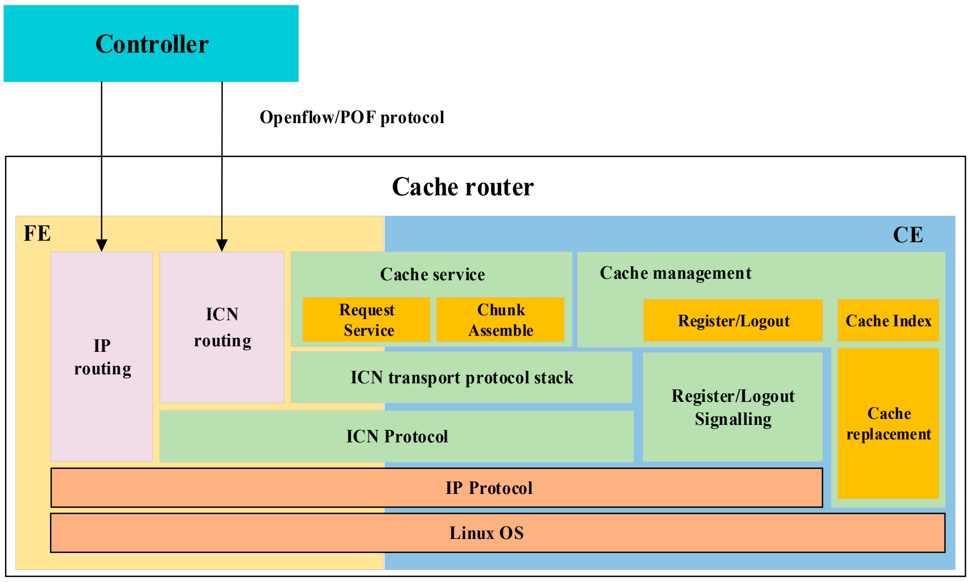

Figure 3. Software architecture of the proposed information-centric networking (ICN) router.

4.2. Implementation

Figure 4 shows a complete implementation architecture of the ICN router in a multi-core environment of commercial servers. Caching and forwarding modules are decoupled into two independent processes. The communication between two processes is completed through shared hugepage memory. The CE that is designed to achieve the cache capacity up to TBs employs the hierarchical framework described in [26]. In order to reduce the overhead of synchronization between threads, each function module is deployed on a dedicated core (thread) that has its own exclusive data structure so that it can perform read/write operations alone.

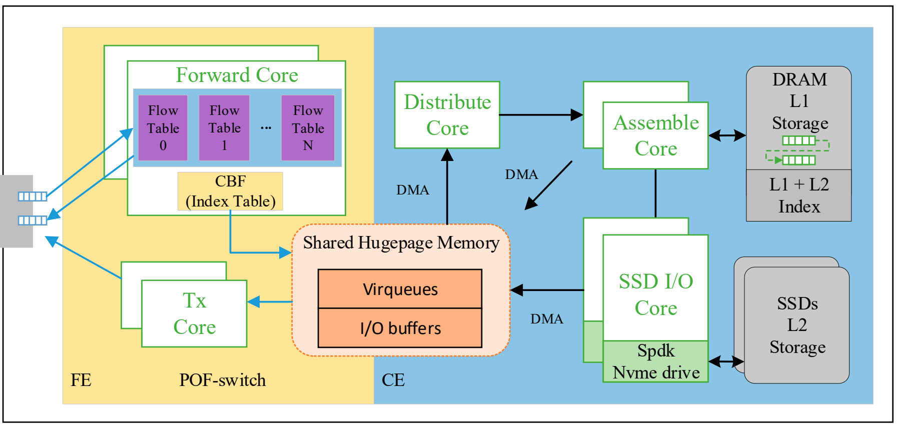

Figure 4. Implementation of the cache-supported ICN router.

4.2.1. Packet Processing Flows

Firstly, we briefly summarize the packet processing flows in our implemented ICN router. When a data packet arrives, the Forward core looks up the counting bloom filter (CBF) to see whether the corresponding chunk has been cached. If not, the packet is sent to the CE by putting it on the virtual queue while being forwarded according to its flow rules in flow table. To achieve the load balance of flows among threads, the Distribute core in the CE distributes packets to assemble cores for chunk assembling afterwards. After the chunk has been assembled and stored in the disk, the CE notifies the TX core to send a register message to the NRS. Similarly, if a chunk is deleted due to the cache replacement policy, the TX core also needs to send a logout message.

When a request packet is received, if the destination address points to the IP address of the router, the requested chunk must be cached in the router. Therefore, the packet is directly sent to the CE; otherwise, it is forwarded according to the flow table. If the request hits the L1(DRAM)/L2(SSD) level caching, the assemble cores or SSD I/O cores place the chunk on the I/O buffer queue via direct memory access (DMA). Finally, the TX core sends the chunk encapsulated with the ICN and IP header out from the port that the request packet comes from.

4.2.2. Forward Element

The FE is mainly responsible for routing and forwarding the received IP and ICN packets. We deployed an open source POF software switch on the forward core to implement the *packet routing* module. We did not need to add CCN-like data structures like Pending Interest Table (PIT) and (Forwarding Information Base) FIB [9] in the POF, and, as consequence of that, the NRS decouples content addressing from data forwarding. We could code the customized routing strategy according to our needs, and then the controller becomes able to issue the corresponding flow table to the router.

We extended POF switch from following three aspects:

1. In order to save precious memory resources, the router should not send ICN packets to the CE when the corresponding chunk has been cached. Therefore, we maintained an index table (IT) for name of cached chunks in POF. Considering the advantage of the CBF that the time of inserting and searching is independent of the number of entries in the table, we implemented the IT via the CBF.
2. We added the new POF instruction *outmulti (port, queue)* in a data plane based on the instruction *outport (port,)* which forwards packets out from *port*. If the FE decides to cache the ICN packet, *outmulti* (*port*, *queue*) puts a copy of the packet on the virtual *queue* between processes while forwarding the packet to the *port*. This instruction also can be used for the realization of the multicast function of ICN.
3. POF cannot maintain line speed forwarding under a data-intensive environment because many CPU cycles are consumed in switching between the kernel and user modes of the Linux

network protocol stack. We optimized the forwarding performance of POF through Data Plane Development Kit (DPDK) technology [38], which accelerates network packet processing by exposing the handling memory area of packet to user-space for DMA with zero-copy.

Moreover, to reduce memory copies and avoid the delay caused by reliable transmission, we implemented the *request response* submodule in the TX core. The TX core polls I/O buffers. If any chunk is placed on the queue, the TX core sends it to the ICN transport protocol stack.

4.2.3. Cache Element

In the cache element, we employed a small but fast DRAM (Layer 1) combined with multiple large but slow SSDs (Layer 2) to construct a hierarchical cache structure that has been proven to be an effective method to extend the cache capacity of an ICN router to the terabyte scale [27].

Dual Queue Management of DRAM

We implemented a chunk assemble submodule in the assemble core. In addition to assembling chunks, it also serves as the L1 cache of the router and manages the L1 and L2 cache indexes. Considering the inevitable delay of assembling chunk, we employed dual queues of first input first output (FIFO) and LRU to manage chunks for leveraging the L1 cache space and efficiently assembling space (as shown in Figure 1). The FIFO queue manages the chunks being assembled, and the LRU queue manages the chunks that have assembled and evicted from the FIFO queue. When the FIFO queue is full, the chunk that started assembling first are deleted, no matter whether they were successfully assembled or not, because we believe that the time an incomplete chunk lives in FIFO is sufficient for assembling.

Optimal Chunk Assembling

The same chunk may be simultaneously transmitted by multiple sources on same path in the network. Hence, packets of the same chunk may reach the router out of order, and the length of each payload packet is not uniform due to the different MTU size settings. To reduce the delay of chunk assembling, we built an optimal scheme in which packets with the same name are put into the same memory area for chunk assembling. In other word, the same chunk from different sources can be assembled at the same time.

To settle the performance problem caused by the frequent dynamic memory allocation, we allocated a large amount of assembling space that is equal to the chunk size in advance. The Assemble cores decapsulate the payload from the primitive packet and put it in assembling space based on payload size and start-offset field in the ICN header, which is a necessary field we should add in ICN header. This process is managed by a double linked list. When a linked list is created, an initial node that records the starting and ending addresses of the assembling memory space is created. A new node is created when the payload of the received packet is filled into the assembling space. Immediately, both nodes are updated to record the starting and ending addresses of two memory spaces separated by payload.

Figure 5 shows four situations that will be encountered during assembling in a pre-malloc memory space. The shaded portion indicates that this space has been filled with data.

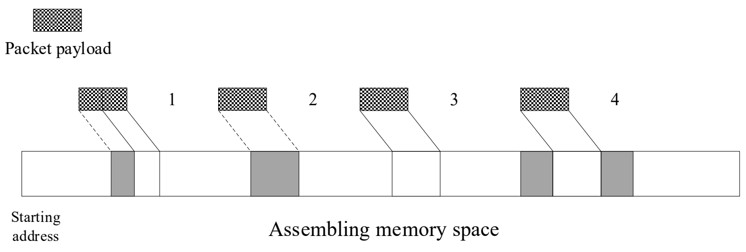

Figure 5. The four situations in process of assembling.

Situation 1: Some part of this space that the payload should fill is occupied.

Situation 2: The space that the payload should fill has already been occupied completely.

Situation 3: Both the space that the payload should fill and the adjacent space are free.

Situation 4: The space that the payload should fill is after an ending, before the starting of occupied data, or in the middle exactly.

Algorithm 1 shows the details how an optimal algorithm of the chunk assembling works. Only if there is only one node left in the linked list and the starting and ending addresses of this node are equal to the addresses of the assembling memory space, a chunk has been assembled completely. This method can improve router memory utilization and the success probability of chunk assembling, especially for popular contents.

Algorithm 1. Chunk Assembling by Packet

```
1. $Offset \leftarrow ReturnPacketOffset(Packet)$
2. $Len \leftarrow ReturnPayloadLen(Packet)$
3. $EID \leftarrow ReturnChunkEID(Packet)$
4. if EID not in the IT then
5.   Create Double linked list
6.   Copy packet payload from $Offset$ to $Offset + Len$
7.   $Node\_count \leftarrow 1$
8. end if
9. if EID in the IT then
10.   if ChunkAssemblingDone(EID) then
11.   Drop Packet
12.   else
13.     for $i = 0$ to $Node\_count$ do
14.       if situation 1 then
15.         Copy packet payload from $EndingAddress(Node[i])$ to $Offset + Len$
16.         $UpdateNode(Node[i], Offset + Len)$
17.       end if
18.       if situation 2 then
19.         Drop Packet
20.       end if
21.       if situation 3 then
22.         Copy packet payload from $Offset$ to $Offset + Len$
23.         $CreateNode(i, Offset, Len)$
24.       end if
25.       if situation 4 then
26.         Copy packet payload from $Offset$ to $Offset + Len$
27.         $UpdateNode(Node[i], Offset + Len)$
28.         if $ChunkIsAssembled(Node[i])$ then
29.           Return Chunk
30.         end if
31.       end if
32.     end for
33.   end if
34. end if
```

Performance Optimization of L2 Cache

SSD I/O cores are responsible for the actual read and write operations in SSDs. The access latency of an SSD up to O(10 us) [28] becomes the performance bottleneck of the CE. However, we argue that a reasonable chunk size in a cache system is up to a few MBs, which is much larger than the size of one SSD page (4–32 KB). Therefore, we paid more attention to improving the throughput of the sequential read and write operations of the SSD. An effective way to avoid a bottleneck is parallel optimization [29]. Therefore, we bound an independent SSD for each SSD I/O core. Multiple SSDs read and write without locking at the same time, which allows for load balance and breaks the throughput limit of a single SSD.

To further improve the IO bandwidth and reduce the access latency of the SSD, we replaced the traditional kernel device driver with a user mode driver named the Storage Performance Development Kit (SPDK) [39]. The SPDK, recently created by Intel, provides a user mode driver for non-volatile memory express (NVMe) SSD device, which avoids kernel context switching and interruption. The non-interrupted polling method saves a large amount of CPU overhead and allows for more instruction cycles to focus on data storage. In addition, the reason we chose an SSD as the L2 cache was because the performance of SSDs has been greatly improved with the emergence of NVMe technology. For example, the read bandwidth has increased from 500 to 3200 MB/s, and the write bandwidth has increased from 400 to about 1200 MB/s. Moreover, the price of SSDs has fallen dramatically from 0.68 to 0.15 $/GB in recent years [40].

5. Performance Evaluation

5.1. Performance Evaluation of ICN Hierarchical Cache System

5.1.1. Simulation Setup

We implemented an ICN HCS based on a hierarchical NRS on a simulation platform named Icarus [41], which is a lightweight simulator that allows users to easily test customized caching and routing policies. Table 1 shows the fundamental parameters we tested in our experiment. A simple binary tree topology was chosen to verify the accuracy of our modeling of the HCS we discussed in Section III. TISCALI, an ISP-like topology depicting the realistic internet environment of Europe, was used to evaluate the performance of HCS under different caching strategies. Each cache node (indexing content in chunk level) deployed LRU as the default replacement policy. We divided the topology into a hierarchical nested structure through a graph-partitioning-based algorithm named latency-aware hierarchical partitioning (LHP) [32]. In order to facilitate the gathering of statistics of signaling overhead in the HCS, we created some new nodes to deploy distributed RNs and bound them to specific attachment nodes chosen by the above algorithm. Both stationary and YouTube Video traffic in real world were used as the tested traffic workload. The stationary workload extracted from a catalog of $N = 1 \times 10^6$ content objects followed the Zipf distribution ($\alpha = 0.8$). The requests of this workload were designed up to 2×10^6 with the IRM model. The YouTube Video traffic records total 3.8×10^6 of requests [42] for 1.76×10^6 of videos, coming from 31,124 distinct IP addresses. Half of the total requests were used to warm up cache system, and the remaining were used to gather statistics. Each experiment in same scenario was independently run ten times, and we analyzed the results with a 95% confidence interval.

Table 1. Simulator environment. LRU: least recently used.

Parameter	Value
Traffic Workload	Stationary/YouTube Video traffic
catalogue size	$1 \times 10^6 / 1.76 \times 10^6$
Topology	Tree, TISCALI
Cache size	Uniform
Replacement	LRU
Request number	$2 \times 10^6 / 1.76 \times 10^6$
Request distribution/rate	Possion/10req/s

5.1.2. Verifying the Accuracy of Analysis Model of ICN Hieratical Cache System

To verify the accuracy of the model of the HCS under different caching strategies (LCD, LCE, and LCP with $p = 0.2$), we implemented three simple ICN hieratical cache systems in a full binary tree (depth = 4). We deployed L3 RN, i.e., the GNRS, at the root node to manage the location information of all content in the topology. The L2/L1 RN, which served for the L2/L1 HEA containing the attachment nodes and the subtrees owned by the child nodes, was deployed at the second/third layer nodes of the tree. In the case of tree-like networks, the cache hit ratio of some upper level nodes could be evaluated step-by-step starting from the leaves and going up to this node through previous Equations (2) and (5)–(7). Then, the cache hit average ratio of the L1/L2 HEA could be calculated by simple summation from the hit ratio of a single node.

Figure 6 reports the average hit ratio of the L1/L2 HEA in different caching strategies, under the two considered approximations, against simulation results. We observed that with the increase of nodes in HEA and cache size, the standard Che approximation tended to overestimate the average hit probability, essentially as a consequence of a large overestimate of the hit probability on every single node. Our optimal Che approximation considered the prediction of request behavior, which brought the analytical prediction of the average cache-hit probability very close to the simulation results. Furthermore, we could obtain a minimum error against simulation results in LCD since the optimized Che approximation we created for LCD was based on a strong assumption that excluded many unknown possibilities. We noted that the biggest gap between the approximation results of the standard Che approach and the simulation results happened in LCE. This was because LCE cached multiple same copies in the same HEA, which resulted in an increase of the number of the overestimated nodes.

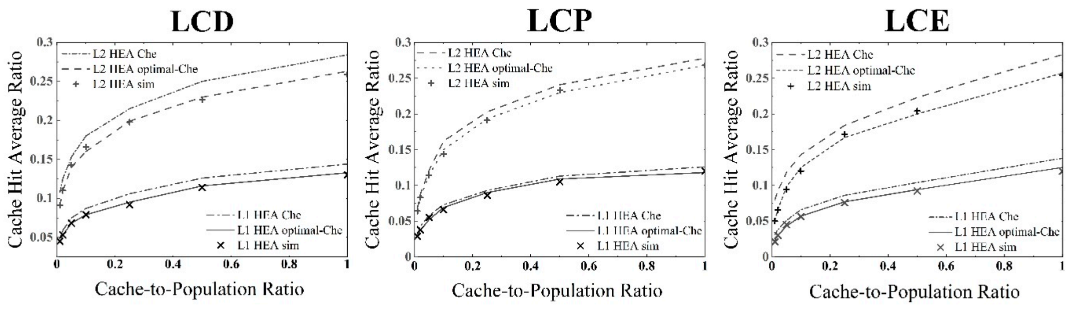

Figure 6. Hit ratio vs cache size for various caching strategies in different approximation approaches. LCD: leave copy down; LCP: leave copy with probability; and LCE: leave copy elsewhere.

5.1.3. Performance Comparison in Real-World ISP Topology

The TISCALI topology has 240 nodes and 810 edges, including 36 content sources, 160 cache-supported routing nodes, and 44 receivers. Three kinds of cache systems were evaluated in different caching strategies (LCE, LCD, CL4M, and LCP with $p = 0.2$). In detail, the first is a HCS based on a hierarchical NRS; the second is a flat cache system (FCS) with a single resolution node (which is a special case of a hierarchical caching system with the single level); and the last is a CCN-like [23] cache system based on routing requests by name.

Figure 7 reports the access latency comparison of different cache systems with different source selection mechanisms (nearest and random) under the workload of YouTube traces from the campus network. With the increase of the routers' cache size, selecting the nearest content copy in the HCS (HCS-N) significantly reduced the download latency among all the caching strategies by up to 18.6% compared with the cache system based on routing requests by name. This proves that, while routing requests by name avoids resolution latency, it is difficult to find an optimal content duplicate. Moreover, the HCS performed better than the FCS because the HCS took advantage of the characteristic of hierarchical content addressing, in which the distance information between users and content is hidden in the hierarchical structure. In other words, the HCS achieved a more efficient content discovery at the cost of a tolerable resolution latency.

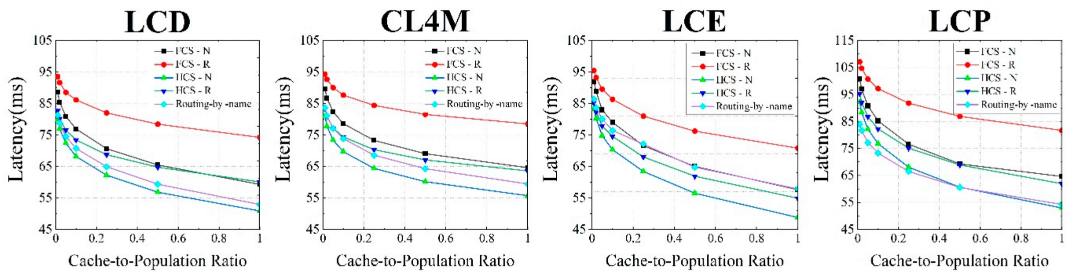

Figure 7. Comparison of latencies with cache-to-population ratios {0.01, 0.02, 0.05, 0.1, 0.25, 0.5, and 1}.

Randomly selecting the content copy in the HCS (HCS-R) achieved the second best performance of access latency, similar to the HCS-N under LCE and LCP. This was because both caching strategies only needed single request to fetch the cache content in the L1 HEA that was closest to user. Contrary to the above strategies, LCD and CL4M only cached one or a few content copies on the transmission path each time, which means that users could not retrieve content in the L1 HEA the first time. This conclusion also confirms our analysis regarding the source selection mechanism in Section 3.2.

As for the cache hit ratio, Figure 8 illustrates that the mechanism of choosing the nearest content copy achieved the highest cache hit rate, which was irrelevant to the structure of a cache system and caching strategies. The HCS-R still achieved a suboptimal performance because of the aforementioned analysis. The performance of the cache systems of routing requests by name in LCD outperformed the other strategies since it was an implicit cooperative caching strategy. It should be noted that LCP achieved a significantly better performance than other strategies because it better avoided the simultaneous placement of the content in all routers.

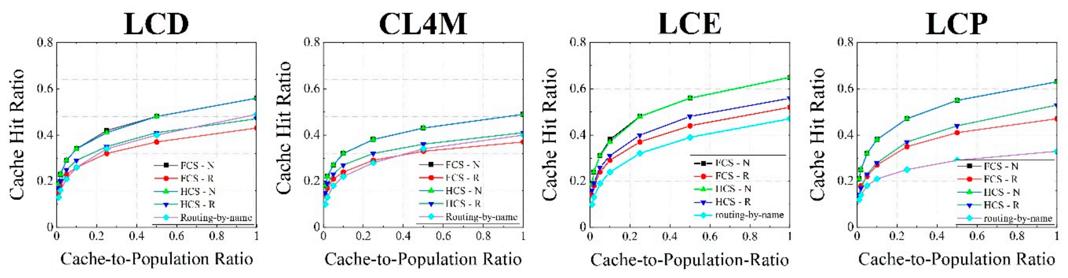

Figure 8. Comparison of cache-hit ratios with cache-to-population ratios {0.01, 0.02, 0.05, 0.1, 0.25, 0.5, and 1}.

5.1.4. Control Overhead

The extra control overhead in a cache system brought from the mechanism of content addressing based on the NRS includes the signaling overhead of resolution, register, and logout. We estimated the impact of control overhead in terms of link load, which is the average number of packets transmitted on a link per second. Figure 9 reports a comparison of average link load in an overall network between three caching systems and different caching strategies. We observed that the hierarchical cache system saved more network bandwidth than the others did, because it helped users find content copies faster. However, LCE was a special case whose control overhead was much higher than the other strategies due to each node in the network actively participating in caching at the same time.

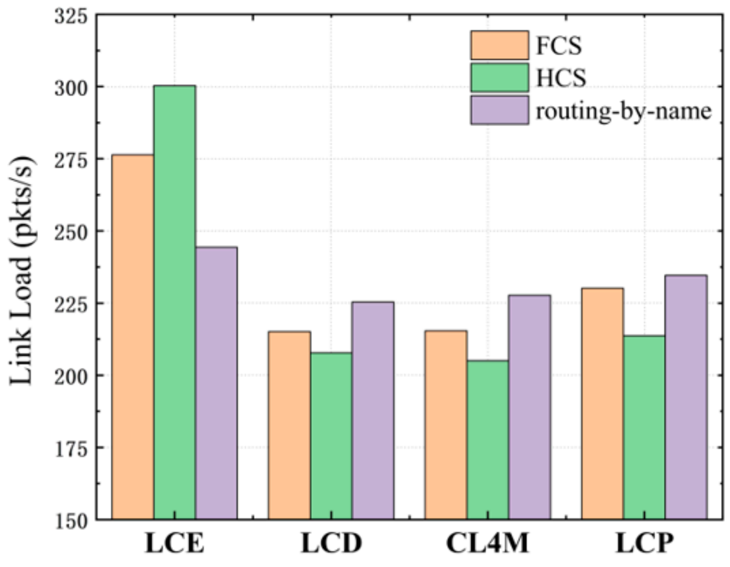

Figure 9. Average link load in overall network.

As shown in Figure 10, we estimated the average access load of each level RN in different cache systems (HCS and FCS). We noted that the RN under the LCE strategy faced 243% more service traffic than the other strategies, while the difference between the loads of other strategies was small. This result showed that we should avoid caching a large amount of copies in one transport process. Moreover, the HCS lessened the service pressure of the L3 RN by directing the service traffic to different level RNs.

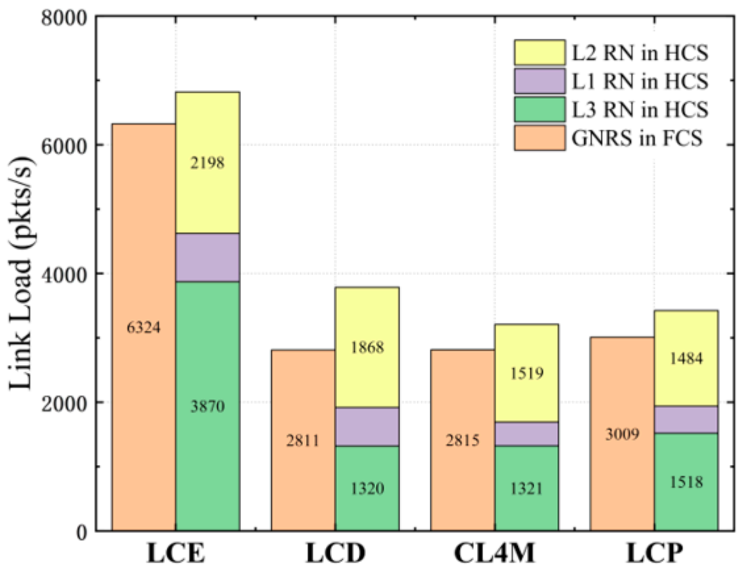

Figure 10. Average access load of each level's resolution node.

5.2. Performance Measurement for Cache-Supported Router

The performance measurement of a router mainly focuses on forwarding and caching. A simple testbed was built from two general-purpose servers, each equipped with two gigabit Ethernet network interface cards (NICs). One server ran an ICN traffic generator [43] and a data sink. The other ran our implemented cache-supported ICN router. The details of the hardware architecture are reported in Table 2. Flow rules were configured and installed on the forward core of the router in advance.

Table 2. Hardware configuration for experimental server. NICs: network interface cards; SSD: solid state disc.

CPUs	2 Intel(R) Xeon(R) CU E5-2620 v3 @ 2.40 GHz
NICs	4*Intel 82599 (10 Gbps)
SSD	Intel Series 750 480 GB*3

To measure forwarding performance, traffic was originated at the traffic generator and transmitted to the router. After the router finished its forwarding operations, packets were sent back to the generator. Figure 11 reports the experimental results. The throughput of the original POF switch was no more than 1 Gbps when the size of test packet was set to 1500 Bytes, which means that it could not meet the requirement of high throughput up to O (10 Gbps) of routers in future networks. However, with the growth of packet size, we could achieve a 10 Gbps throughput on the POF optimized by DPDK in the FE, i.e., forwarding on a 10 GbE network card with a line speed. Furthermore, we did not need to worry about the impact of small size packets on forwarding because the chunk size in ICN would not be smaller than MTU, such as 4K in CCNx.

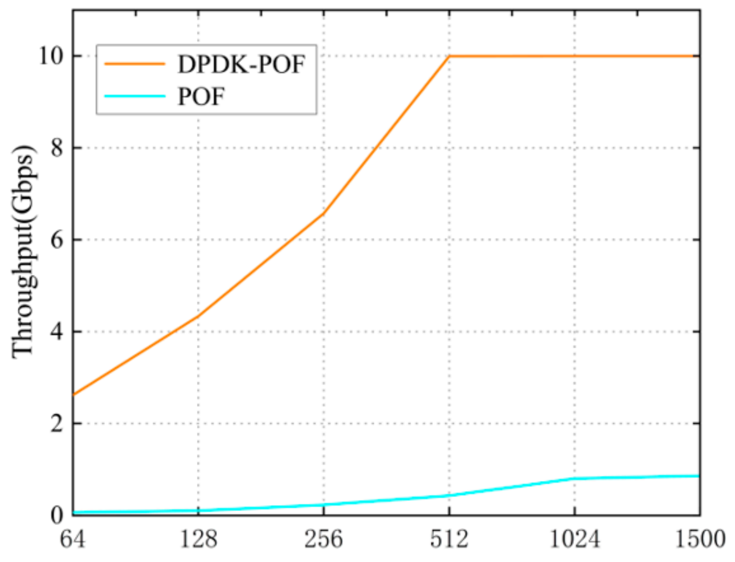

Figure 11. Packet forwarding throughput.

As for caching performance, we only concentrated on the performance improvement brought from user mode drive rather than the process of assembling because it was affected by too many factors, such as packet loss rate and chunk transmission rate. In addition, due to the distribute core distribute packets according to the hash of content name, the irregular flows could have led to the unbalanced loads in some assemble cores, which could have also had a negative impact on performance.

Two kinds of cache-supported routers were implemented by applying different device drives: the SPDK of the user mode and Vector Input/Output (VIO) [29] of the kernel mode. We measured the throughput of a single SSD over 10 runs of 100,000 data chunks with different sizes while the queue depth of the SSD was set to 4. Figure 12 reports the result as a function of read/write mix under two implementation schemes. We noted that the throughput of the SSD based on the SPDK of the user mode drive was much better than that based on the kernel model drive of Linux. With the increase of the chunk size, the performance rapidly improved because the IO operation pattern changed from random to sequential. We argue that reasonable chunk sizes can be up to few MBs since the IO capacity of a single SSD is sufficient to cover 10 Gbps of traffic. We then conclude that an NVMe SSD promises to be the main medium in the storage of cache-supported ICN routers, as it removes the IO throughput bottleneck of multi-Gbps ICN routers.

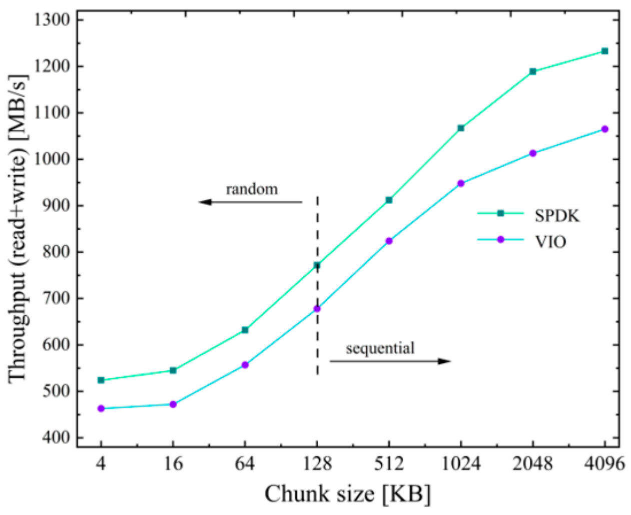

Figure 12. Throughput comparison of different block device drives.

6. Conclusions

In this paper, we discussed the ICN cache system based on a hierarchical NRS as an HCS according to the summary of the behavior pattern of request. We firstly designed a simple three-layer HCS compatible with IP. The standard Che approximation for modeling a single cache node could not be extended to the network because it ignored the correlation of requests from adjacent nodes. We then proposed an optimal approach to overcome the weakness of Che approximation, and we applied it to model and analyze the hit ratio of different levels of the HCS. Secondly, to achieve the incremental deployment of the HCS in an IP network, we proposed a complete design of a cache-supported ICN router. Considering the influence of MTU on chunk assembling, an optimal chunk assemble algorithm was designed to mitigate this problem. We then implemented it with a high performance in a parallel environment.

Experimental results showed the accuracy of our analysis in the modeling of the ICN cache system under different caching strategies. The HCS could reduce the service pressure of the server and download delay more than a flat cache system that forwarded packets based on name. The hierarchical structure attached potential location attributes to content addressing, which indicated that the content addressing achieved by name resolution was more efficient than by discovering in the routing process; meanwhile, both had similar traffic overheads in the network. The measurements of our implemented router demonstrated that the performance of the forwarding and IO operations of caching can be improved by some dedicated acceleration frameworks (DPDK and SPDK) working in the user mode, which makes it possible for an ICN router to satisfy the requirement of a high line rate. Moreover, the bottleneck of a router moves from SSD access time to the latency of the assemble process when the size of the chunk is set to large. We hope that this paper can provide ideas for more research on the modeling, design, and implementation of ICN cache systems in the future.

Author Contributions: Conceptualization, L.Z., H.N., and R.H.; methodology, L.Z., H.N.; software, L.Z.; writing—original draft preparation, L.Z.; writing—review and editing, L.Z., H.N. and R.H.; supervision, H.N.; project administration, R.H.; funding acquisition, H.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by Strategic Leadership Project of Chinese Academy of Sciences: SEANET Technology Standardization Research System Development (Project No. XDC02070100).

Acknowledgments: We would like to express our gratitude to Hong Ni, Rui Han, and YuanHang Li for their meaningful support for this work.

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

References

1. Din, I.U.; Kim, B.-S.; Hassan, S.; Guizani, M.; Atiquzzaman, M.; Rodrigues, J.J.P.C. Information-Centric Network-Based Vehicular Communications: Overview and Research Opportunities. *Sensors* **2018**, *18*, 3957. [CrossRef] [PubMed]
2. Vahlenkamp, M.; Schneider, F.; Kutscher, D.; Seedorf, J. Enabling ICN in IP networks using SDN. In Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP), Goettingen, Germany, 7–10 October 2013; pp. 1–2.
3. Raychaudhuri, D.; Nagaraja, K.; Venkataramani, A. MobilityFirst: A robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **2012**, *16*, 2–13. [CrossRef]
4. Dannewitz, C.; Kutscher, D.; Ohlman, B.; Farrell, S.; Ahlgren, B.; Karl, H. Network of Information (NetInf)—An information-centric networking architecture. *Comput. Commun.* **2013**, *36*, 721–735. [CrossRef]
5. Wang, G.Q.; Zheng, Q.; Ravindran, R. Leveraging ICN for Secure Content Distribution in IP. In Proceedings of the Networks. The 7th ACM International Conference on Multimedia System (MMSys), Klagenfurt amWörthersee, Austria, 10–13 May 2016; pp. 765–767.
6. Al-Khalidi, M.; Thomos, N.; Reed, M.J.; Al-Naday, M.F.; Trossen, D. Network controlled mobility management using IP over ICN architecture. *CoRR* **2016**. Available online: https://www.researchgate.net/profile/Nikolaos_Thomos/publication/309551505_Network_Controlled_Mobility_Management_using_IP_over_ICN_Architecture/links/5897c4bd92851c8bb67f0890/Network-Controlled-Mobility-Management-using-IP-over-ICN-Architecture.pdf (accessed on 2 September 2020).
7. Trossen, D.; Reed, M.J.; Janne, R.; Georgiades, M.; Fotiou, N.; Xylomenos, G. IP over ICN—The better IP? In Proceedings of the European Conference on Networks and Communications (EuCNC), Paris, France, 29 June–2 July 2015.
8. Jacobson, V.; Smetters, D.K.; Diana, K.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking named content. In Proceedings of the 5th international conference on Emerging networking experiments, Rome, Italy, 1–4 December 2009; pp. 1–12.
9. Zhang, L.; Alexander, A.; Jeffrey, B.; Jacobson, V.; Claffy, K.; Crowley, P.J.; Papadopoulos, C.; Wang, L.; Zhang, B. Named data networking. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
10. Liao, Y.; Sheng, Y.; Wang, J. A Temporally Hierarchical Deployment Architecture for an Enhanced Name Resolution System. *Appl. Sci.* **2019**, *9*, 2891. [CrossRef]
11. Hong, J.; Chun, W.; Jung, H. A flat name based routing scheme for information-centric networking. In Proceedings of the 2015 17th International Conference on Advanced Communication Technology (ICACT), Seoul, Korea, 1–3 July 2015; pp. 4444–4447.
12. Koponen, T.; Chawla, M.; Chun, B.-G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A data-oriented (and beyond) network architecture. In Proceedings of the ACM SIGCOMM Computer Communication Review, Kyoto, Japan, 27 August 2007; pp. 181–192.
13. Thomas, Y.; Xylomenos, G. Towards improving the efficiency of ICN packet-caches. In Proceedings of the 10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (Qshine), Rhodes, Greece, 18–20 August 2014; pp. 186–187.
14. Laoutaris, N.; Che, H.; Stavrakakis, I. The LCD interconnection of LRU caches and its analysis. *Perform. Eval.* **2006**, *63*, 609–634. [CrossRef]
15. Laoutaris, N.; Syntila, S.; Stavrakakis, I. Meta algorithms for hierarchical Web caches. In Proceedings of the IEEE International Performance, Computing and Communications Conference, Phoenix, AZ, USA, 15–17 April 2004; pp. 445–452.
16. Chai, W.K.; Diliang, H.; Ioannis, P.; George, P. Cache "less for more" in information-centric networks. In Proceedings of the International Conference on Research in Networking, Berlin/Heidelberg, Germany, 21–25 May 2012; pp. 27–40.
17. Saino, L.; Ioannis, P.; George, P. Hash-routing schemes for information centric networking. In Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking, Hong Kong, China, 12 August 2013; pp. 27–32.
18. Naeem, M.A.; Ali, R.; Alazab, M. Enabling the content dissemination through caching in the state-of-the-art sustainable information and communication technologies. *Sustain. Cities Soc.* **2020**, *61*, 102291. [CrossRef]

19. Meng, Y.; Naeem, M.A.; Ali, R. EHCP: An Efficient Hybrid Content Placement Strategy in Named Data Network Caching. *IEEE Access* **2019**, *7*, 155601–155611. [CrossRef]
20. Che, H.; Tung, Y.; Wang, Z. Hierarchical Web caching systems: Modeling, design and experimental results. *IEEE J. Sel. Areas Commun.* **2002**, *20*, 1305–1314. [CrossRef]
21. Sofman, L.B. Analytical model of hierarchical cache optimization in the network with unbalanced traffic. In Proceedings of the 20th Communications & Networking Symposium, Baltimore, MD, USA, 23–26 April 2017; pp. 1–12.
22. Fricker, C.; Philippe, R.; James, R. A versatile and accurate approximation for LRU cache performance. In Proceedings of the 24th International Teletraffic Congress (ITC), Kraków, Poland, 4–7 September 2012.
23. Martina, V.; Garetto, M.; Leonardi, E. A unified approach to the performance analysis of caching systems. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM), Toronto, ON, Canada, 27 April–2 May 2013.
24. Ioannis, P.; Richard, G.C.; Raul, L. Modelling and Evaluation of CCN-Caching Trees. In Proceedings of the International Conference on Research in Networking, Berlin/Heidelberg, Germany, 9–13 May 2011; pp. 78–91.
25. Perino, D.; Varvello, M.; Linguaglossa, L.; Rafael, B.R. Caesar: A content router for high-speed forwarding on content names. In Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Los Angeles, CA, USA, 10–21 October 2014; pp. 137–148.
26. Ding, L.; Wang, J.L.; Sheng, Y.Q.; Wang, L.F. A Split Architecture Approach to Terabyte-Scale Caching in a Protocol-Oblivious Forwarding Switch. *IEEE Trans. Netw. Serv. Manag.* **2017**, *14*, 1171–1184. [CrossRef]
27. Saino, L. On the Design of Efficient Caching Systems. Ph.D. Thesis, College London, London, UK, 2015.
28. Rossini, G.; Rossi, D.; Garetto, M.; Leonardi, E. Multi-terabyte and multi-Gbps information centric routers. In Proceedings of the IEEE Conference on Computer Communications(INFOCOM), Toronto, ON, Canada, 27 April–2 May 2014; pp. 181–189.
29. Won, S.; Taejoong, C.; Yuan, H.W.; David, O.; Mark, S. Toward terabytescale caching with SSD in a named data networking router. In Proceedings of the 10th ACM/IEEE Symposium on Architure for Networking Communication System, Los Angeles, CA, USA, 20–21 October 2014; pp. 241–242.
30. Zafar, H.; Abbas, Z.H.; Abbas, G.; Muhammad, F.; Tufail, M.; Kim, S. An Effective Fairness Scheme for Named Data Networking. *Electronics* **2020**, *9*, 749. [CrossRef]
31. Stefano, S.; Andrea, D.; Matteo, C.; Pomposini, M.; Blefari-Melazzi, N. Transport-layer issues in information centric networks. In Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking, Helsinki, Finland, 17 August 2012; pp. 19–24.
32. Li, J.; Sheng, Y.; Deng, H. Two Optimization Algorithms for Name-Resolution Server Placement in Information-Centric Networking. *Appl. Sci.* **2020**, *10*, 3588. [CrossRef]
33. Liao, Y.; Sheng, Y.; Wang, J. A Brief Survey on Information Centric Networking Proof of Concepts for IMT-2020 and Emerging Networks. *J. Netw. New Media* **2018**, *7*, 54–63.
34. Elisha, J.R.; Jim, K.; Don, T. Approximate Models for General Cache Networks. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM), San Diego, CA, USA, 14–19 March 2010; pp. 1–9.
35. Feamster, N.; Jennifer, R.; Ellen, Z. The road to SDN: An intellectual history of programmable networks. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–97. [CrossRef]
36. Song, H.Y. Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 13–16 August 2013; pp. 127–132.
37. Pat, B.; Dan, D.; Glen, G.; Izzard, M.; McKeown, N.; Rexford, J.; Schlesinger, C.; Talayco, D.; Vahdat, A.; Varghese, G.; et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–95.
38. Intel DPDK Framework. Available online: http://dpdk.org/ (accessed on 5 May 2016).
39. Storage Performance Development Kit. Available online: http://www.spdk.io/ (accessed on 28 May 2018).
40. Yang, Z.Y.; Harris, J.R.; Walker, B.; Daniel, V.; Liu, C.; Chang, C.; Cao, G.; Stern, J.; Verma, V.; Paul, L.E. SPDK: A Development Kit to Build High Performance Storage Applications. In Proceedings of the IEEE International Conference on Cloud Computing Technology & Science, Hong Kong, China, 11–14 December 2017; pp. 154–161.

41. Saino, L.; Psaras, I.; Pavlou, G. Icarus: A caching simulator for information centric networking (ICN). In Proceedings of the Seventh International Conference on Simulation Tools and Techniques, Lisbon, Portugal, 17–19 March 2014; pp. 66–75.
42. Michael, Z.; Kyoungwon, S.; Gu, Y.; Kurose, J. Watch global, cache local: YouTube network traffic at a campus network: Measurements and implications. *Multimed. Comput. Netw.* **2008**, *6818*, 681805–681813.
43. Ostinato Network Traffic Generator and Analyzer. Available online: http://ostinato.org/ (accessed on 11 April 2010).