future internet

MDPI

Article

T-Move: A Light-Weight Protocol for Improved QoS in Content-Centric Networks with Producer Mobility

Swaroopa Korla [1,*] and Shanti Chilukuri [2,*]

[1] RISE Krishna Sai Gandhi Group of Institutions, Ongole 523272, India
[2] GITAM Institute of Technology, Gandhi Institute of Technology And Management (Deemed to Be University), Visakhapatnam 530045, India
* Correspondence: swaru2004@gmail.com (S.K.); chilukurishanti@gmail.com (S.C.)

Received: 21 November 2018; Accepted: 24 January 2019; Published: 27 January 2019

Abstract: Recent interest in applications where content is of primary interest has triggered the exploration of a variety of protocols and algorithms. For such networks that are information-centric, architectures such as the Content-Centric Networking have been proven to result in good network performance. However, such architectures are still evolving to cater for application-specific requirements. This paper proposes T-Move, a light-weight solution for producer mobility and caching at the edge that is especially suitable for content-centric networks with mobile content producers. T-Move introduces a novel concept called *trendiness* of data for Content-Centric Networking (CCN)/Named Data Networking (NDN)-based networks. It enhances network performance and quality of service (QoS) using two strategies—cache replacement and proactive content-pushing for handling producer mobility—both based on trendiness. It uses simple operations and smaller control message overhead and is suitable for networks where the response needs to be quick. Simulation results using ndnSIM show reduced traffic, content retrieval time, and increased cache hit ratio with T-Move, when compared to MAP-Me and plain NDN for networks of different sizes and mobility rates.

Keywords: content-centric networking; quality of service; edge caching

1. Introduction

Recent advances in microelectronic and mechanical devices and wireless communication have given rise to large networks of mobile nodes for various purposes. These nodes may now be sensors, actuators, vehicles, and humans. Some of these are constrained in terms of energy, processing power, memory, etc. In addition, there has been a clear shift in interest towards data based on its content, rather than the host of the content. Gathering, processing, and communicating data in such large networks of mobile devices needs special architectures and protocols. In addition, service and device discovery and routing of data is a major challenge in applications with mobile nodes such as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) networks. Solutions for service discovery and mobility that work in the Internet (such as DNS and Mobile IP) are not efficient enough in their current form for such networks because of the highly dynamic nature of content. For example, Mobile IP requires each device to have a unique IP address, which may not be possible in the case of many devices, whose life span may be very short. 6LoWPAN (based on IPV6) makes this possible to some extent, but service/device discovery remains a problem.

One way to solve the addressing and routing problem is information-centric networking (ICN). ICN is based on the fact that most users are interested in the data—not the address at which it is located. In the current Internet architecture, routing of the request for a resource (the Uniform Resource Identifier, URI) and the reply (e.g., the file named by the URI) is done solely based on the IP address of the source and destination computers. In ICN, the user requests the resource using a *name* similar to the URI. Requests are routed using longest prefix-matching of the URI, and the content is routed to

the user on the reverse path. As devices do not have addresses, resolution of services to addresses is not necessary. MobilityFirst [1], which is another popular ICN architecture, uses both resource names and node addresses. The resource name is called the Global Unique IDentification(GUID) and is very similar to the URI. The difference between the GUID and the URI is that the GUID does not contain the address like the URI does. Hence, data can be routed based on the address or name. Routing schemes and tables in MobilityFirst are more complex than those in ICN.

This paper proposes T-Move, based on Content-Centric Networking (CCN [2,3]) and Named Data Networking (NDN [4]). CCN is a popular ICN architecture which makes use of in-network caching. NDN is derived from CCN and has the same core network elements and data structures. T-Move is built on just these basic network elements and data structures and hence can be used with both NDN and CCN implementations. Routers in CCN/NDN maintain a *content store* (CS) to cache data and satisfy further interests for this data. In-network caching is an excellent mechanism to reduce the data retrieval time and network traffic. Both these are very important parameters in the context of many devices generating huge amounts of data. Hence, an information-centric approach has been proposed and evaluated successfully for several applications ([5,6] etc.). However, the cache replacement policy [7] which decides what cache items are stored, plays a major role in the performance of caching. In the context of networks with mobile devices and low latency requirements, a cache replacement policy that can improve system performance is much more difficult to design. CCN/NDN inherently performs well even when consumers are mobile, as data objects are *pulled* by consumers and follow a path set by the interest. producer mobility, on the other hand, is not easy to handle [8], as producers cannot be located with IP-like addresses in CCN/NDN.

There are three main approaches to handle mobility in CCN/NDN—anchor-based, rendezvous-based, and anchorless. All these approaches are costly in terms of time and number of updates. Besides these, proactive content-pushing [9,10], where the producer proactively pushes data to the content router even if no interest is received, is a popular solution to send a continuous stream of data even if a producer is moving. Again, pushing all content to all neighboring routers proactively may not be possible in networks with large amounts of data (for example, audio and video streaming in V2X networks).

In this paper, we propose T-Move, which is a producer mobility solution for content-centric applications with edge caching. T-Move uses a novel metric called the *trendiness* of a data object, which is indicative of the popularity of similar content. T-Move contains two separate strategies—one for cache replacement and another for (producer) mobility management. Both these strategies are simple and especially suitable for applications that require low response times. In T-Move, when a producer moves, a copy of its content is maintained at an edge router that is most likely to receive an interest for it and Forwarding Information Base (FIB) entries of routers are updated with minimum update messages to point to the new location of the producer or the cached content. Simulation results show that T-Move results in reduced traffic and increased cache hit ratio, even when the content producers are mobile, leading to better QoS than MAP-Me, an anchorless producer mobility solution for content-centric networks [11].

The rest of the paper is organized as follows: Section 2 explains the caching and mobility management policies in CCN, which is a popular ICN architecture. Section 3 deals with work related to cache replacement policies and mobility management in the web and ICN. Section 4 discusses the T-Move policy proposed by us. Simulation results using ndnSIM are presented in Section 5 and we conclude the paper in Section 6.

2. Caching and Mobility Support in the CCN Architecture

In this section, we discuss the basic caching scheme used in information-centric networks. We illustrate this using a popular ICN architecture, CCN [2] by PARC. The basic network elements in an ICN are:

- Producers—network nodes which are producers of data.
- Consumers—network nodes which request for data objects. Typically, producers and consumers are end user devices.
- Edge routers—routers that are directly connected to the end devices. These routers connect the producers and consumers to the core network.
- Network routers—routers that form part of the core network.

Routers (both edge and network) in CCN/NDN maintain three important data structures—the CS, the Pending Interest Table (PIT) and the FIB. Consumers of content express their interest for a data object in the form of an *interest packet*. A router which receives the interest packet tries to retrieve the content object from its CS. In case the data object is not present in its CS (i.e., a cache miss occurs) and the object is not already listed in the PIT, the router adds the interface of the incoming interest packet to its PIT and the interest packet is sent out on an appropriate out-going face according to the FIB. If the data object is present in the CS, the router sends it to the consumer and does not forward the interest packet. As the data object travels to the consumer, intermediate routers delete the corresponding entry in the PIT and store the data object in their caches using First-In, First-Out (FIFO) replacement policy. For a detailed description of CCN, please refer to [2].

To support mobility, CCN uses announcements by the sources of data (producers) [2]. Producers perform a Register operation which announces the prefixes they service to the CCN core. Upon receiving the announcement, all CCN routers create local FIB entries for the registered prefixes of the announcer and the face of the announcer. The registered prefixes use flags to indicate if they should be announced further. The advertisements might be via CCN, via standard IP Service Location protocols, or via CCN or IP routing [2].

3. Related Work

3.1. Caching in Information-Centric Networks

Caching has been widely explored in the form of web caching to reduce the content retrieval time. The caching decision policy decides when and where data must be cached and is a major aspect of caching. Two extreme solutions to the caching decision problem are the *cache-everything* scheme, where all data through a router is cached by it, and the *cache-nothing*, where no data is cached at the intermediate nodes.

An important issue in caching is the cache replacement policy to be followed. There are several contending solutions for this ranging from simple Least Recently Used (LRU) and FIFO to complex popularity-based learning algorithms. LRU, FIFO or random replacement schemes may serve fairly well if the data objects are uniform in size and priority. However, it is an established fact that frequency-based cache replacement schemes out-perform LRU and other simple schemes [12].

While predicting the popularity of online content has been studied by several researchers ([13,14] etc.), using it for caching is a relatively recent trend. In [15], the authors study how popular social media content can be used to optimize content replication. In [16], the authors use an off-line model-based scheme for forecasting popularity and model parameters are obtained using training datasets. The issue with training-based schemes is that the popularity of content changes with time and models based on older training sets may not result in correct prediction of popular content.

More recently, Ref. [17] has proposed PopCaching which works online. The novelty of PopCaching is that it does not require a training phase and learns about the popularity of content online. PopCaching converges quickly and shows promising results in terms of cache hit ratio. However, PopCaching follows a greedy approach based on the number of hops for deciding where content must be cached. This caches popular content closer to the user and unpopular content closer to the servers and implies that two routers at the same hop distance away from the requesting users have the same probability of caching the content. In reality, however, such routers may not receive interests for data with the same probability.

Caching in ICN has been the topic of interest for many researchers. Popular ICN designs and projects (CCN, DONA, PSIRP, NETInf) categorized caching into on-path and off-path caching [18,19]. On-path caching minimizes the content retrieval time and bandwidth consumption by caching the content at intermediate nodes on that path from producer to consumer, but results in high content replication. Off-path caching focuses more on frequency of access of the content and cache capacity to allocate cache space optimally [19]. ICN inherently supports caching in either form, but has different features and requirements compared with the IP-based Internet, as pointed out by [12]. Except for a few, all existing web caching solutions cache entire data objects. On the contrary, data in ICN is partitioned in sub-objects or chunks. This may result in chunks of the same content object being cached at different routers. Also, routers in ICN must make caching decisions and replacements online, which makes most web caching solutions unsuitable for ICN [20].

In a related study, Ref. [20] proposes random fractional caching for CCN. In this scheme, the routers maintain a packet store and an index table to locate each packet. Each packet has a pre-set probability of getting indexed and cached. Our argument is that packet caching and replacement should not be random, but should be based on the popularity of the content, instead of being random. Refs. [21–23] present popularity-driven caching strategies for ICN. These solutions formulate the caching problem as an optimization problem and/or use coordinated caching for replicating data. However, coordination requires time and results in increased traffic in the core network. In [24], the authors propose an ICN-based caching scheme for improving retrieval time in 5G networks with mobile video consumers. They do not consider mobility of producers (videos are normally stored in servers in the core network) and use LRU for cache replacement.

When the CCN architecture is used for devices which require low latency and/or mobile, the cache decision policy needs to be simpler and more efficient. Ref. [12] studies the performance of various caching decision schemes for content-centric networks using a realistic YouTube-like catalog. The major take-away of this study is that simple schemes out-perform complex ones and popularity of content plays a crucial role in the performance of a cache replacement policy. Accordingly, T-Move proposed in this paper is a simple scheme that works online and caches content based on its trendiness. This leads to content being cached at routers where it is most likely to be accessed, irrespective of the hop-count or geographical distance.

3.2. Producer Mobility in Information-Centric Networks

Very little work has been reported on producer mobility for CCN/NDN. The authors in [18] survey various techniques to handle mobility in information-centric networks. As discussed in Section 1, there are three major approaches to producer mobility—anchor-based, rendezvous point-based, and anchorless. Anchor-based solutions (e.g., [25–27]) have special network nodes called *anchors* that keep track of the mobile producer and reroute interests to its current location. Rendezvous-based approach (e.g., [28–31]) works in a similar way where a rendezvous node redirects interests to the mobile producer. The major difference between these two approaches is that the anchor is a node that belongs to the home domain of the producer, whereas the rendezvous point may be anywhere in the network.

The disadvantage of the above approaches is that the anchor/rendezvous point becomes a network performance bottle neck. Also, failure of an anchor/rendezvous point results in loss of information about current content location. Anchorless approach to producer mobility does not need require special nodes such as anchors or rendezvous points. Instead, special messages (called interest updates) are sent to the routers, which change their FIB entries for the mobile content to reflect the new face on which interests must be forwarded to reach the producer's new location. Ref. [29] proposes one such anchorless scheme where the FIB entries are updated, while more recently, Ref. [32] proposed MobiCCN, which uses greedy routing to handle mobile producers. Each node in the network is assigned a virtual coordinate and when a producer moves it sends a greedy update to a special router called its *host router*. All the routers along the path of this update update the FIB entry to be the face

on which the update is received. This allows them to send future interests for that content to the new location. MobiCCN performs well, but requires special nodes acting as host routers and virtual coordinates to be assigned to nodes. MAP-Me [11] has been proposed to provide anchorless producer mobility in information-centric networks. MAP-Me uses Interest Updates by mobile producers to update the FIB entries. None of the above approaches to mobility leverage the fact that content can be retrieved not only from the producer, but also from the routers that cache it.

Our mobility support solution is based on proactive content-pushing, as it ensures uninterrupted access to data. Proactive content-pushing to a nearby router when the producer moves has been studied in [9,10]. In these papers, content is pushed to a nearby router even when the producer receives no interest for it. Since content gets cached at the router, it can be retrieved from the router's CS if the producer moves away. While [9] gives no simulation results, [10] considers a very small network with 9 routers and just one producer and one consumer with 10 interests/s and a mobility rate of 15 m/s for evaluation. In [33], the authors propose a scheme that pushes content proactively to increase availability and evaluate it for larger topologies. However, content is pushed upon creation in any network other than the home network. As mentioned in Section 1, proactively pushing content that may never be requested unnecessarily increases traffic and adversely effects cache memory management at routers.

T-Move is an anchorless, proactive content-pushing solution in the sense that it does not require special nodes. It goes a step further from anchorless solutions that merely update the FIB entries to reroute interests—it pushes content (based on trendiness) at routers where it is most likely requested from and sends FIB updates so that the content *or* its cached copy can be found by interests. This decreases the retrieval time of content. T-Move also reduces the number of FIB updates by dropping them wherever possible, thus decreasing the traffic substantially in mobile environments such as vehicle-to-infrastructure networks.

4. T-Move

T-Move is built on CCN [2]/NDN [4], but improves system performance by modifying two policies of CCN/NDN—the cache replacement strategy and the mobility support strategy. The idea is to use prefix-matching to measure the popularity of content and use this for cache replacement and efficient mobility support. We describe the two important modules of T-Move in the sections below. In this discussion, we use the term edge router to specifically denote edge routers connected to the producers of data. Other routers along the route from this edge router to the consumer (network routers and edge routers connected to the consumer) are called *upstream routers*.

4.1. Trendiness-Based Cache Replacement Policy in T-Move

Interest forwarding in T-Move is done as in the normal CCN architecture. Routers in T-Move (and CCN) maintain three important data structures—the CS, the PIT and the Forwarding Information Base (FIB). Upon receiving a request, a router (say R1) sends a copy of the data object from the CS or forwards the interest, depending on whether the object is cached at it or not. In the latter case, some other router along the path or the producer itself may reply with the data object. The data object travels along the reverse path (following the PIT entries) and reaches router *R1*. Following on-path caching, *R1* now caches the data object.

The important aspect that impacts the system performance here is the cache replacement policy followed by the router. Plain (vanilla) CCN uses FIFO for this. As discussed in Section 3, frequency and popularity-based cache replacement strategies show significant boost in performance for web caching. Web requests follow a Zipf distribution and the Zipf popularity distribution for content class k when there are N content classes is defined by $q_k = \frac{c}{k^{/}alpha}$. The normalized value of c is $\sum_{k=1}^{N} \frac{1}{k^{\alpha}}$, where α is the *skewness of content popularity*. It has been shown that a cache hit ratio of about 70% can be obtained even when 2% to 7% of the web pages are stored for α ranging between 2 and 1.2 [17].

This is very encouraging for in-network caching of popular content. However, existing off-line or online popularity-based caching schemes for content-centric networks are not suitable to networks with high rates of mobility and data rates (Section 3).

In T-Move, we leverage the naming hierarchy of CCN and propose a simple but effective cache replacement policy. This is based on the fact that Zipf's Law can be applied not only to requests for single web documents, but for "strides of requests", i.e., a sequence of requests from a client [34]. When hierarchical naming is employed, this means that the demand for data with similar prefixes is higher. For example, if the producer has a web site with pages *a/b/c/p1, a/b/c/p2, a/b/c/p3 ...*, once a page is requested, there is a high likelihood of the rest of the pages being requested. If a video named *p/q/r/*, is requested, it may be returned as chunks named *p/q/r/c1, p/q/r/c2 ....* . Our policy uses prefix-matching and frequency for determining content that is currently in *trend*. Each router in T-Move maintains a *Trendiness Table TT*, which has three entries per data object in the router's cache—the name of the data object, the number of times it was requested (its *frequency*, denoted by f) and its *"trendiness"*, denoted by t as shown in Figure 1. The trendiness of a data object is calculated based on the normalized frequency of access of *similar* data objects in the name hierarchy using prefix-matching as below:

$$t_j = \frac{\Sigma_{i=1}^{n} p_i * f_i}{P_j} \tag{1}$$

where n is the number of data objects in the cache of the router, P is the number of prefixes of the j-th data object, p_i is the number of prefixes that match in the j-th data object and the i-th data object and f_i is the frequency of access of the i-th data object. For example, considering a data object with name *x/y/z/c1*, P_j is 4, as the name has four prefixes. Let a router have three data objects whose names match with this data object in some way. If this router's cache has a data object d_1 with frequency of access f_1 and name *x/y/z/c1*, the p_1 value is 4, as there are four prefixes that match in the names. For a second data object d_2 with name *x/y/z/c2* cached in the router, p_2 is 2, as two prefixes match. Let the frequency of d_2 be f_2. If the router has cached a third data object with name *x/u/v/c1* and frequency f_3 and p_3 is 1. The trendiness of data object *x/y/z/c1* is obtained by $t_1 = (4f_1 + 3f_2 + f_3)/4$. The trendiness hence takes into account not only the popularity of the data object, but also that of related content. The more closely two data objects are related to one another, the more prefixes match in their names, which increases their impact on the trendiness of each other. The trendiness is normalized with respect to the number of prefixes in the data object for which it is calculated. In the absence of such normalization, data objects with longer prefixes maybe reported to be trendier even if the frequency of access of related objects is less. The trendiness of all data objects with at least one prefix-matching is updated in the TT when a data object is stored or accessed from the cache.

Cache replacement is carried out based on the trendiness of the data object. When a router receives a data object, it stores a copy in its cache and forwards the object to the consumer. If there is space in the cache without having to delete anything, the new data object is simply inserted into the cache and an entry into is made in the TT with its name, trendiness, and frequency. If there is no place in the cache, the data object with least trendiness is evicted from the cache to store the new data object. Cache replacement based on trendiness is a simple, yet effective means of measuring the popularity of content. It is dynamic and with low overhead, which makes it an attractive option for reducing traffic and latency.

4.2. Mobility Support in T-Move

A major problem with information-centric networks is the mobility of producers (or providers) of content. To illustrate this, we use a simple network where the producer and consumer are separated by two routers R_1 and R_2, such that the route from the producer to consumer is *producer-R1-R2-consumer*. Let the interests for a particular data object d_i arrive at router R_2 as a Poisson arrival rate of λ per second and the router's cache hold N data objects at a time. Assuming that interests for data objects other than d_i arrive at a rate of μ per second, authors of [35] give an expression for the rate of interest

propagation from one router (R_1) to the next router (R_2) as $\lambda(\frac{\lambda}{\lambda+\mu})^N$. If the transmission range of the router R_1 is R_{tx} meters and the producer moves at a uniform rate of vm/s, it stays within the router's range for a maximum of $\frac{R_{tx}}{v}s$. After this time, all the interests sent by R_1 to the producer expire.

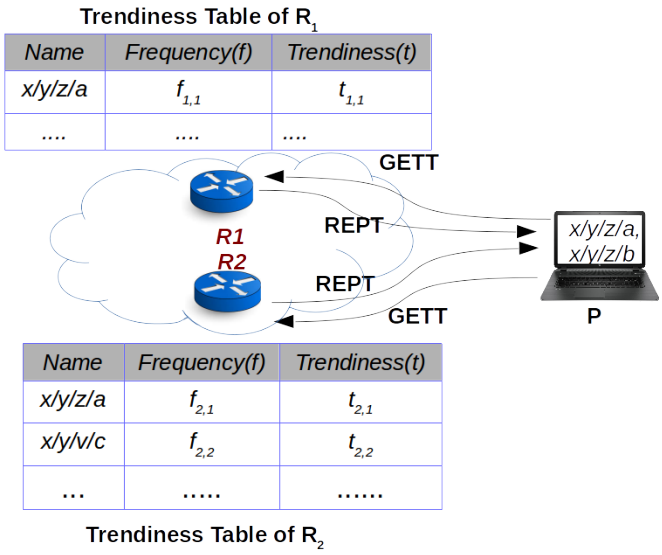

Figure 1. Example scenario for producer mobility.

Traditional IP-based schemes using tunneling and FIB updates have been proposed by some authors to handle producer mobility in ICN. However, most of them just deal with a way of rerouting interests to the producer that has moved. Since ICN uses in-network caching, we propose a caching decision policy to store the content of a mobile producer in the cache of a nearby *favored* router with minimal FIB updates to route interests to this router. A favored router is an edge router that has content similar to the one at the producer in its cache. It has been shown by many researchers that content exhibits spatio-temporal popularity [34] and hence, caching content where similar content is frequently accessed gives better network performance. Storing data objects at favored routers results more interests being satisfied by the cached content, resulting in a higher cache hit ratio and lower retrieval time, as presented in Section 5.

4.2.1. Producer Behavior

Consider a content producer P with the two data objects *x/y/z/a* and *x/y/z/b*, and is initially connected to edge routers R1 and R2, as in Figure 1. Initially, the data is only at the producer and not cached at any router. Eventually, R1 and R2 get interests for *x/y/z/a* separately, get the data object from P and store it in their cache. When further interests for the data object are received, R1 and R2 just return the data from their caches and increment the frequency and trendiness of the data object. Let these values for data object *x/y/z/a* be ($f_{1,1}$, $t_{1,1}$) and ($f_{2,1}$, $t_{2,1}$) at routers R1 and R2 respectively. In addition, let R2 also have data object *x/y/v/c* in its cache, with frequency and trendiness values ($f_{2,2}$, $t_{2,2}$).

When P starts moving away within the same domain, it broadcasts a control message *GETT* (short for *GET Trendiness*) with the names of data objects it has. Each edge router receiving this message does the following for each data object listed in the *GETT* message:

- It checks if that data object is already in the cache. If so, it discards the *GETT* and does nothing else. In Figure 1, both R1 and R2 do nothing for data object *x/y/z/a*.

- If the data object is not cached, the router calculates the trendiness of the data object returns this value to P on the reverse path in a *REPT* (short for *REPort Trendiness*) message. In the above example, R1 and R2 return the trendiness of *x/y/z/b*, because it is not cached.

The mobile producer P collects all the *REPT* messages from all its neighboring routers and sends a copy of the data object to be cached at the router that reports highest trendiness for that data object (say R2 in the above example). The data object is now cached at the router (R2) and the producer. This behavior of a producer is described in Algorithm 1.

Algorithm 1 Proactive_data_push (D)

1: /*Run by a producer moving away from the edge router*/
2: /*D is list of data objects at the producer*/
3: Send $GETT(D)$
4: Collect $REPT(D)$ for τ seconds
5: Choose router R_i that reports maximum trendiness for each data object d_i
6: Send each d_i to the corresponding R_i
7: /*All data objects are sent to edge routers that report maximum trendiness*/

4.2.2. Behavior of Edge Routers

When the location of the data object changes, the FIB entries of edge and upstream routers may need to change to route interests properly. Interests for a data object may be routed to the producer or a router that has a cached copy. This is achieved by viewing the path to a producer as two sub-paths—one from the consumer to the edge router and the other from the edge router to the producer. To forward interests from the edge router to the producer, the edge routers behave optimistically. If a router (e.g., R1 and R2 in Figure 2) sends a REPT, it is on the path to the producer, though temporarily. It adds the face on which the *GETT* was received to its FIB for this data object. This results in FIB entries to the producer's data at both the old and new edge routers. To facilitate interest forwarding on the first part of the path (from the consumer to the edge router), FIB update messages are sent from downstream to upstream routers to inform them of changes to be made in the FIB. Two types of FIB updates are possible—those requesting deletion of FIB entries and those requesting addition of FIB entries.

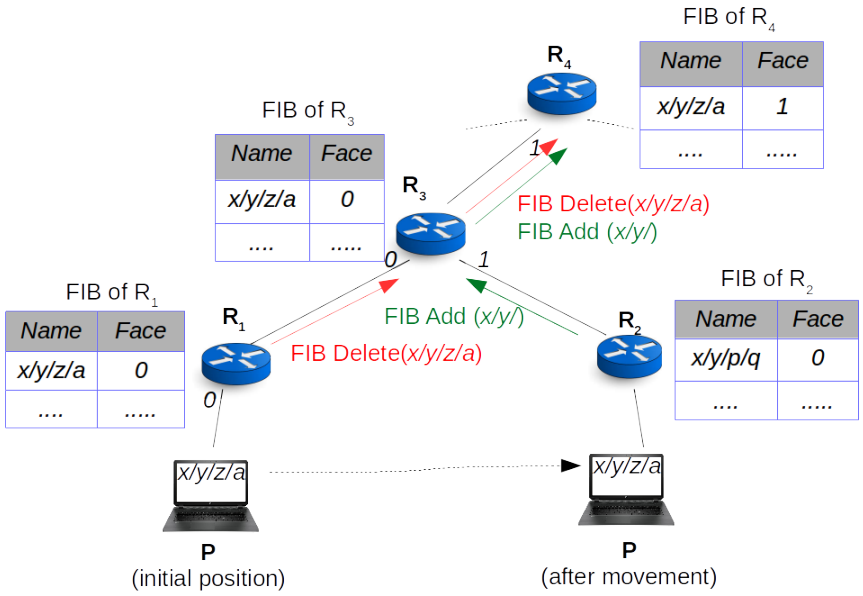

Figure 2. FIB updates to common upstream router (Case 1).

A router that receives a $GETT$ message from a producer is connected to the producer. It calculates the longest prefix match string (*x/y/* in the example) between the new data object in the $GETT$ message and those in its cache and sends a FIB_Add update with this string to its upstream router (R3 in Figure 2). This makes it possible for future interests to be routed to both R1, which is on the path to the producer before movement and R2, which on the path after movement. The functioning of an edge router upon receiving a $GETT$ message is described in Algorithm 2. Please note that only one of the edge routers is the new edge router connected to the producer and one of them has cached data as it is the favored edge router.

Algorithm 2 Cache_trendy_data(TT,FIB,D,CS)

```
/*Run by an edge router upon receiving a GETT packet*/
/*TT is trendiness table, FIB is the Forwarding Information Base and CS is the content store of the
router; D is the set of data objects received in the GETT(D) packet*/
for each data object d_i in D do
    if d_i is not in CS then
        /* d_i is not already cached at the router*/
        Set trendiness value of d_i from TT in REPT(D)
    end if
end for
Send REPTD if there is at least one d_i is not cached
for each data object d_i in D do
    /*Cache d_i if possible, send FIB updates to upstream routers*/
    if data object d_i is received within γ seconds then
        Add d_i to CS
    end if
    Determine string s_i from FIB with longest prefix-matching with d_i
    Send FIB_Add(s_i) packet to upstream router R_u
end for
/*Data objects are cached at favored routers, FIB entries made at edge routers and FIB updates
sent upstream*/
```

Two cases arise depending on the location of caching and the producer:

- the favored router is the producer's new edge router. This is on-path caching, as caching is done on the path to the location of the producer.
- the favored router is not the producer's new edge router. This results in *off-path caching*.

We discuss these two cases in detail below, keeping in mind that interests for a data object may be satisfied by the producer or a router that has the cached copy.

- *On-path Caching*: If the favored router is the producer's new edge router, FIB_Add updates are enough to route the Interests to the new edge router. If the edge router has a copy of the data requested, it returns the copy. If this copy is no longer in its cache, it forwards the interest to the producer based on its FIB entries. Anyway, interests are routed to the data.
- *Off-path Caching*: While the producer always has the data, a router that has a cached copy may evict the copy from its cache in due course of time. FIB_Add messages cause FIB entries to be added to the cached copy, as well as the actual location of the data. However, in case the data is eventually evicted from the cache of the favored router that is off-path, the FIB entries of upstream routers no longer point to the content.

To deal with this, when a (edge or network) router receives an interest for which it has no cached data or FIB entry, it sends an FIB_Delete message on the reverse path. Upon receiving such a message, upstream routers delete the FIB entry for that content and face, and send the FIB_Delete message further upstream. This removes all the wrong FIB entries pointing to a router (off-path) that has no cached data.

4.2.3. Behavior of Upstream Routers

Upstream routers follow Algorithm 3 to deal with FIB updates with minimum messaging. This algorithm is discussed below in detail.

Algorithm 3 Update_FIB(TT, FIB, CS, s_i)

/*Run by an upstream router R_u upon receiving a $FIB\_Add(s_i)$ and/or $FIB\_Delete(s_i)$ from a downstream router R_d*/
/*TT is trendiness table, FIB is the Forwarding Information Base and CS is the content store of R_u; s_i is the string that matches the longest prefix of a data object sent by R_d*/
if both $FIB\_Add(s_i)$ and $FIB\_Delete(s_i)$ are received **then**
    f_{add} = face on which the $FIB\_Add(s_i)$ packet is received
    f_{delete} = face on which the $FIB\_Delete(s_i)$ packet is received
    if f_{add} not equalsf_{delete} **then**
        Delete f_{delete} for s_i in FIB
        Add f_{add} for s_i in FIB
    end if
end if
if only $FIB\_Add(s_i)$ or $FIB\_Add(s_i)$ is received **then**
    if $FIB\_Add(s_i)$ is received **then**
        Add f_{add} for longest prefix-matching s_i in FIB
    else
        if f_{delete} is the only face for s_i in the FIB **then**
            Delete entry for s_i in FIB
            Send $FIB\_Delete(s_i)$ to upstream router
        else
            Delete face f_{delete} in entry for s_i in FIB
        end if
    end if
end if
/*FIB entries point to data location; both cached and original*/

- *Case 1—Upstream router receives FIB updates for addition and deletion of the same data object*: In this case, the upstream router has a route to the old and new location of data, and it does the following:
  - If it receives addition and deletion updates on different faces (R3 in Figure 2), it deletes the face on which the deletion message is received and adds the face on which the addition message is received. It then drops both the FIB messages. Hence the Face entry for *x/y/z/a* in R3's FIB will change from 0 to 1 in Figure 2.
  - If it receives FIB updates requesting both deletion and addition via the same face, it is connected to both routers on the same face(e.g., R4 in Figure 2). Interests can be forwarded as per the existing FIB entry and still reach the new router R2. R4 just drops the FIB update messages and does nothing.

- *Case 2—Upstream router receives FIB update for addition or deletion of a data object*: In Case 2, upstream routers are not connected to both the old and new location of the data object by any face, such as R3 and R4 in Figures 3 and 4. These figures depict the state of the FIBs after the FIB updates are sent from R1, R2 and after the FIB updates are sent from R3, R4, respectively.
  - If a router receives a FIB update only to *delete* the FIB entry, it checks if the face on which the update is received is the only face listed in the FIB for this data object. If yes, it deletes the entry and forwards the update to its upstream router (R3 in Figures 3 and 4 for *x/y/z/b*). Else, it updates the FIB entry by deleting the face and drops the FIB update (R3 in Figures 3 and 4 for *x/y/z/a*).
  - If it receives a FIB update only to *add* the FIB entry, it adds the face on which the FIB update is received to the longest prefix-matching the string in the update message and drops the update (R4 in Figures 3 and 4 for *x/y/z/a*).

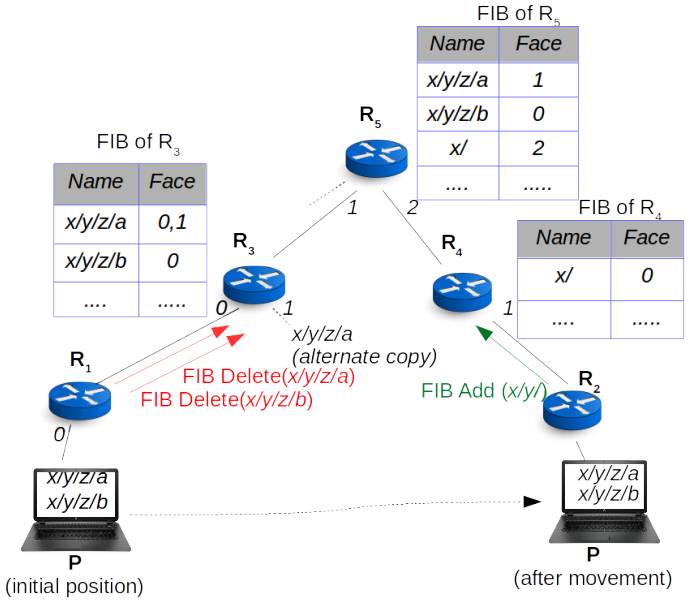

Figure 3. FIB updates from different upstream routers (Case 2—Step 1).

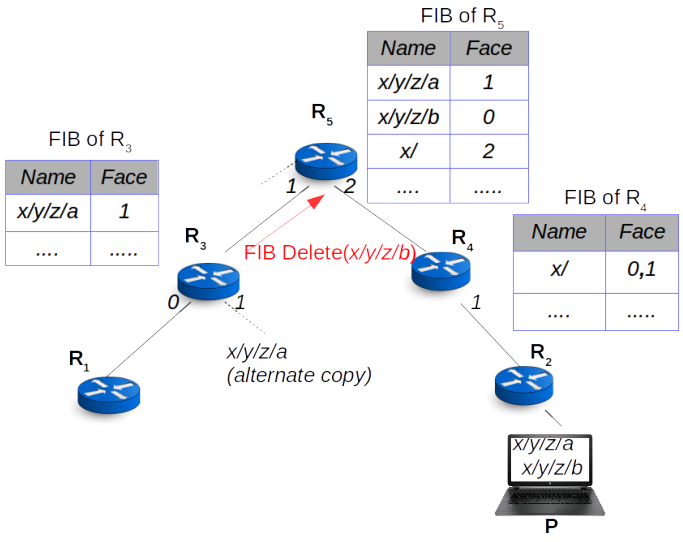

Figure 4. FIB updates from different upstream routers (Case 2—Step 2).

The above scheme reduces the number of FIB updates by dropping the update messages wherever possible. At the same time, interests can still be routed to the new location of data. The new location of data is based on trendiness which is in turn based on prefix-matching. Since prefix-matching is also the basis for interest forwarding, new interests will eventually be routed by upstream routers to reach the caching router, from where the data object can be sent. Since data is cached at a place where it is most likely to be accessed from, the cache hit ratio is better than naive proactive pushing of data, even if it requires some FIB updates. The overall number of messages (and traffic in bytes) exchanged is still less than plain NDN or MAP-Me, which is also an anchorless producer mobility scheme.

5. Simulation Results

To study the effect of T-Move in the CCN/NDN architecture, we simulated a network of nodes in ndnSIM, which is a popular simulator for information-centric networks. We used a random walk mobility model for the nodes in two sample network topologies—Topology 1 with a network of size 50 $\times$ 50 m and Topology 2 with a 250 $\times$ 250 m network. Nodes were distributed randomly in both networks. Interests are generated for 200 data objects, each of size 4 kB according to the Zipf distribution with a popularity exponent of 1.25, as suggested by [12]. Ten interests are generated per second. The interests can be for CBR (e.g., objects in a web page) or non-real-time multimedia (e.g., buffered video/audio) data. Three protocols were simulated (within a confidence interval of 95%)—T-Move , Vanilla NDN, and MAP-Me, which is also an anchorless producer mobility solution.

Figures 5a shows the variation of the cache hit ratio as the number of nodes in the network changes for Topology 1, with a mobility rate of 1 m/s. Figure 5b shows the variation of the cache hit ratio as the number of nodes changes for Topology 2, with a mobility rate of 10 m/s. In both cases, the cache size was of 100 chunks, with a chunk size of 10 KB. When compared with plain NDN (Vanilla NDN), it can be seen that T-Move fares better for both topologies, especially for greater number of nodes. This is because a larger number of nodes increases the number of interests generated which in turn increases the probability of copies of data being cached. We did not consider MAP-Me when comparing the cache hit ratio, as it argues against in-network caching. (MAP-Me argues against in-network caching as this may lead to caching of unpopular content. Since T-Move ensures that only popular content is cached, in-network caching enhances the network performance instead of having a negative impact on it.)

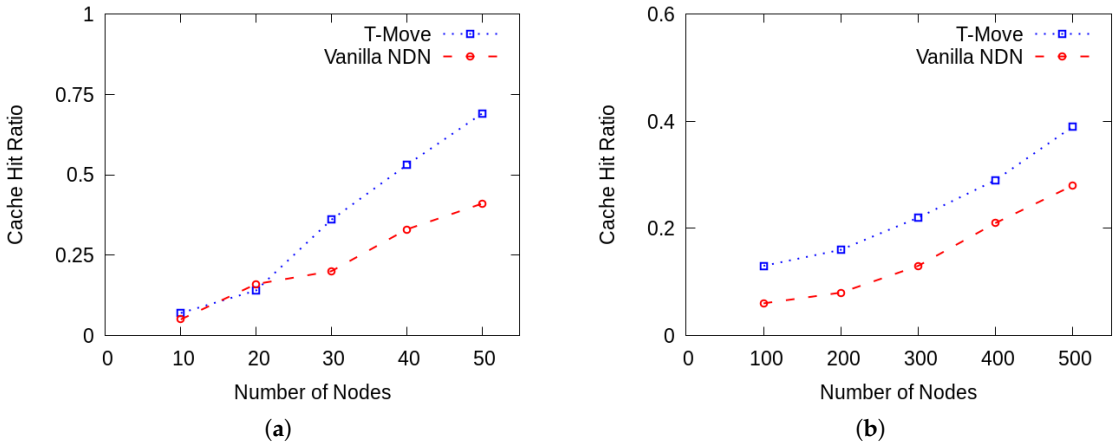

Figure 5. (**a**) Effect of the Number of Nodes on the Cache Hit Ratio (Topology 1); (**b**) Effect of the Number of Nodes on the Cache Hit Ratio (Topology 2).

Figure 6a shows the variation of the average content retrieval time as mobility rate changes, for 50 nodes and a cache of 100 chunks with Topology 1. Figure 6b shows the variation of the average content retrieval time with mobility, for a larger network with 500 nodes and a cache of 100 chunks with

Topology 2. For both topologies, the retrieval time for T-Move is lesser than that for Vanilla NDN and MAP-Me. This is because T-Move differs from Vanilla NDN and MAP-Me in two important ways:

- it caches content based on trendiness and
- it updates FIB entries to point to the new content location or the location of a cached copy.

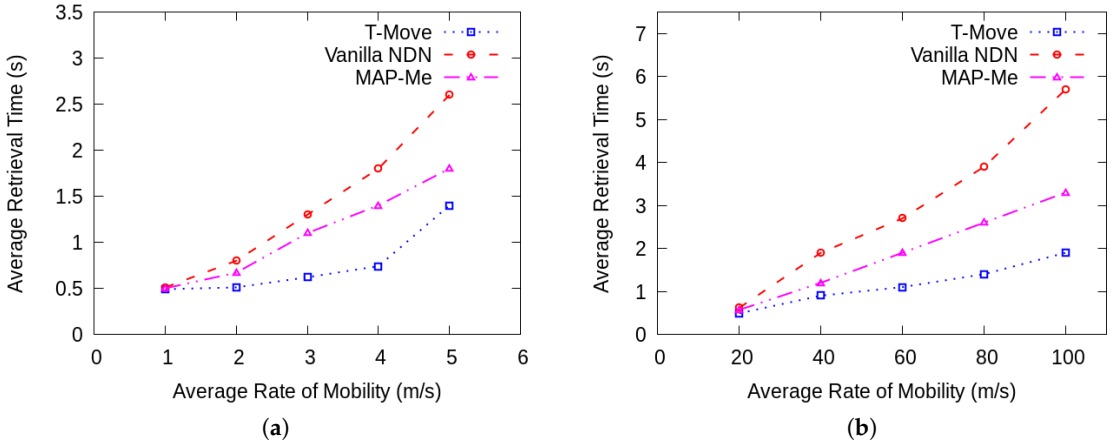

Figure 6. (**a**) Effect of Mobility Rate on Content Retrieval Time (Topology 1); (**b**) Effect of Mobility Rate on Content Retrieval Time (Topology 2).

To study the traffic in the network with T-Move, we considered Topology 1 with a fixed mobility rate of 1m/s and Topology 2 with a mobility rate of 10 m/s and calculated the average traffic in the network in a fixed simulation time and interest generation rate (10 interests/s). For both topologies, the cache size was 100 chunks, with 10 kB per chunk. Due to caching based on trendiness, T-Move results in lesser traffic (Figure 7) compared to Vanilla NDN or MAP-Me for both topologies. This is because with T-Move, data is cached close to where it might be used, in addition to mobility support with minimal FIB updates. This reduces the number of hops taken by interests and data objects to reach the (cached) data object or consumer, respectively. Though T-Move has more control message overhead (in terms of GETT and REPT messages), it performs better than Vanilla NDN or MAP-Me because of the above reason. It can also be noted that as the network size and mobility rate increases, T-move fares much better than Vanilla NDN or MAP-Me.

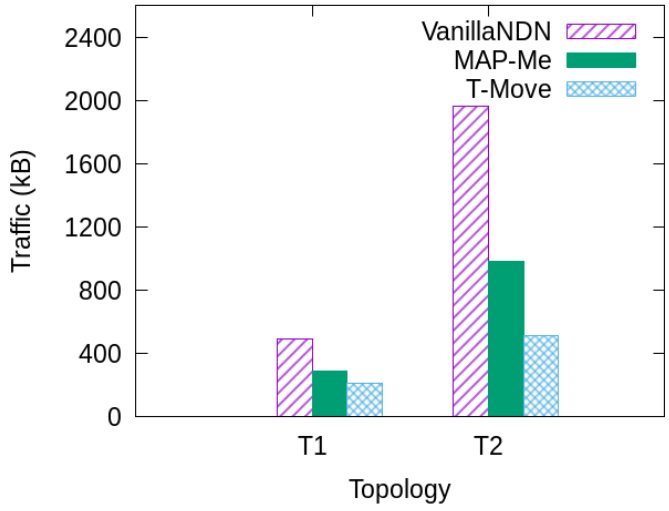

Figure 7. Average traffic for Topologies 1 and 2.

Figures 8a,b show the effect of the cache size on the cache hit ratio in a network with 50 nodes with a mobility rate of 1 m/s with Topology 1 and in a network with 500 nodes moving randomly with 10 m/s with Topology 2. The cache size is varied in chunks of 10 kb each. Because of trendiness-based caching, T-Move results in a better hit ratio compared to Vanilla NDN. The interesting point is that larger networks benefit better from caching due to a greater number of interests, which results in more cached data.

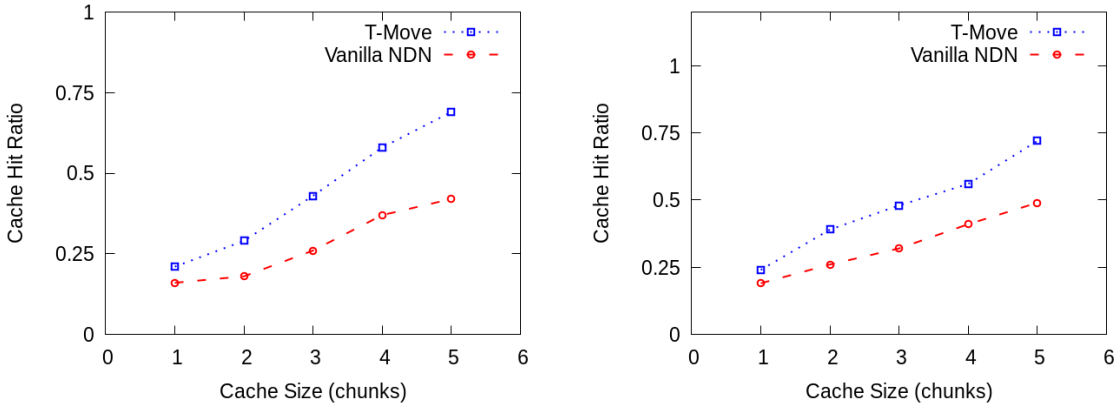

Figure 8. (**a**) Change in Cache Hit Ratio with Cache Size (Topology 1); (**b**) Change in Cache Hit Ratio with Cache Size (Topology 2).

6. Conclusions

In view of the wide range of applications which are mobile and require low latency, we propose T-Move, a caching mechanism for content-centric networks. T-Move uses a novel parameter called *trendiness* of data for caching and handling producer mobility with a light-weight proactive content-pushing module. The cache replacement and content-pushing modules are simple and light-weight, reducing traffic in the core network and latency. Simulation results show increased cache hit ratio, retrieval time and lesser traffic compared to a basic CCN/NDN-like caching policy and MAP-Me. In future, we intend to formulate the cache allocation problem in proactive content-pushing using an optimization formulation.

Author Contributions: Conceptualization, S.K. and S.C.; Methodology, S.K. and S.C.; Software, S.K.; Supervision, S.C.; Validation, S.K.; Writing—original draft, S.K.; Writing—review & editing, S.C.

Acknowledgments: The authors thank the anonymous reviewers for their invaluable suggestions that made this paper more meaningful. The authors also thank Ramasastry Chilukuri for his inputs to the development of the original manuscript.

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

Abbreviations

The following abbreviations are used in this manuscript:

QoS	Quality of Service
ICN	Information-Centric Networking
CCN	Content-Centric Networking
DNS	Domain Name Service
LRU	Least Recently Used
FIFO	First-In First-Out
CS	Content Store
FIB	Forwarding Information Base
PIT	Pending Interest Table

TT	Trendiness Table
GETT	GET Trendiness
REPT	REPort Trendiness

References

1. Venkataramani, A.; Kurose, J.F.; Raychaudhuri, D.; Nagaraja, K.; Mao, M.; Banerjee, S. MobilityFirst: A Mobility-centric and Trustworthy Internet Architecture. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 74–80. [CrossRef]
2. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking Named Content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '09), Rome, Italy, 1–4 December 2009; pp. 1–12.
3. Mosko, M.; Solis, I.; Scott, G.; Walendowski, A. *CCNx 1.0 Protocol Architecture*; Technical report; ACM: New York City, NY, USA, 2015.
4. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named Data Networking. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
5. Shannigrahi, S.; Fan, C.; Papadopoulos, C. Request Aggregation, Caching, and Forwarding Strategies for Improving Large Climate Data Distribution with NDN: A Case Study. In Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN '17), Berlin, Germany, 26–28 September 2017; pp. 54–65.
6. Bouk, S.H.; Ahmed, S.H.; Kim, D. Vehicular Content Centric Network (VCCN): A Survey and Research Challenges. In Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, 13–17 April 2015; pp. 695–700.
7. Che, H.; Tung, Y.; Wang, Z. Hierarchical Web Caching Systems: Modeling, Design and Experimental Results. *IEEE J. Sel. A. Commun.* **2006**, *20*, 1305–1314. [CrossRef]
8. 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]
9. Ko, H.; Kim, Y.; Suh, D.; Pack, S. A proactive content pushing scheme for provider mobility support in information centric networks. In Proceedings of the 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2014; pp. 523–524.
10. Woo, T.; Park, H.; Jung, S.; Kwon, T. Proactive neighbor pushing for enhancing provider mobility support in content-centric networking. In Proceedings of the 2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN), Shanghai, China, 8–11 July 2014; pp. 158–163.
11. Augé, J.; Carofiglio, G.; Grassi, G.; Muscariello, L.; Pau, G.; Zeng, X. MAP-Me: Managing Anchor-Less Producer Mobility in Content-Centric Networks. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 596–610. [CrossRef]
12. Rossi, D.; Rossini, G. Caching Performance of Content Centric Networks under Multi-Path Routing (and More). Available online: https://netlab.pkusz.edu.cn/wordpress/wp-content/uploads/2012/04/Caching-performance-of-content-centric-networks-under-multi-path-routingand-more.pdf (accessed on 20 January 2019).
13. Gürsun, G.; Crovella, M.; Matta, I. Describing and forecasting video access patterns. In Proceedings of the 2011 Proceedings IEEE INFOCOM, Shanghai, China, 10–15 April 2011; pp. 16–20.
14. Szabo, G.; Huberman, B.A. Predicting the Popularity of Online Content. *Commun. ACM* **2010**, *53*, 80–88. [CrossRef]
15. Wang, Z.; Zhu, W.; Chen, X.; Sun, L.; Liu, J.; Chen, M.; Cui, P.; Yang, S. Propagation-based Social-aware Multimedia Content Distribution. *ACM Trans. Multimedia Comput. Commun. Appl.* **2013**, *9*. [CrossRef]
16. Wu, Y.; Wu, C.; Li, B.; Zhang, L.; Li, Z.; Lau, F.C.M. Scaling Social Media Applications into Geo-distributed Clouds. *IEEE/ACM Trans. Netw.* **2015**, *23*, 689–702. [CrossRef]
17. Li, S.; Xu, J.; van der Schaar, M.; Li, W. Popularity-driven content caching. In Proceedings of the IEEE INFOCOM 2016—The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.

18. Tyson, G.; Sastry, N.; Rimac, I.; Cuevas, R.; Mauthe, A. A Survey of Mobility in Information-centric Networks: Challenges and Research Directions. 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. 1–6.
19. Abdullahi, I.; Arif, S.; Hassan, S. Survey on Caching Approaches in Information Centric Networking. *J. Netw. Comput. Appl.* **2015**, *56*, 48–59. [CrossRef]
20. Zhang, G.; Li, Y.; Lin, T. Caching in Information Centric Networking: A Survey. *Comput. Netw.* **2013**, *57*, 3128–3141. [CrossRef]
21. Li, J.; Wu, H.; Liu, B.; Lu, J.; Wang, Y.; Wang, X.; Zhang, Y.; Dong, L. Popularity-driven coordinated caching in named data networking. In Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Austin, TX, USA, 29–30 October 2012; pp. 15–26.
22. Borst, S.; Gupta, V.; Walid, A. Distributed caching algorithms for content distribution networks. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–9.
23. Wang, W.; Sun, Y.; Guo, Y.; Kaafar, D.; Jin, J.; Li, J.; Li, Z. CRCache: Exploiting the correlation between content popularity and network topology information for ICN caching. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, NSW, Australia, 10–14 June 2014; pp. 3191–3196. doi:10.1109/ICC.2014.6883812. [CrossRef]
24. Zhang, Z.; Lung, C.H.; Lambadaris, I.; St-Hilaire, M. When 5G meets ICN: An ICN-based caching approach for mobile video in 5G networks. *Comput. Commun.* **2018**, *118*, 81–92. [CrossRef]
25. Lee, J.; Cho, S.; Kim, D. Device mobility management in content-centric networking. *IEEE Commun. Mag.* **2012**, *50*, 28–34. [CrossRef]
26. Hermans, F.; Ngai, E.; Gunningberg, P. Global Source Mobility in the Content-centric Networking Architecture. 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. 13–18.
27. Jiang, X.; Bi, J.; Wang, Y.; Lin, P.; Li, Z. A content provider mobility solution of named data networking. In Proceedings of the 2012 20th IEEE International Conference on Network Protocols (ICNP), Austin, TX, USA, 30 October–2 November 2012; pp. 1–2.
28. Ravindran, R.; Lo, S.; Zhang, X.; Wang, G. Supporting seamless mobility in named data networking. In Proceedings of the 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012; pp. 5854–5869.
29. Kim, D.-H.; Kim, J.-H.; Kim, Y.-S.; Yoon, H.-S.; Yeom, I. Mobility Support in Content Centric Networks. In Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking (ICN '12), Helsinki, Finland, 13–17 August 2012; pp. 13–18.
30. Zhou, Z.; Tan, X.; Li, H.; Zhao, Z.; Ma, D. MobiNDN: A mobility support architecture for NDN. In Proceedings of the 33rd Chinese Control Conference (CCC), Nanjing, China, 28–30 July 2014; pp. 5515–5520.
31. Tang, J.; Zhou, H.; Liu, Y.; Zhang, H.; Gao, D. A source mobility management scheme in content-centric networking. In Proceedings of the 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2014; pp. 176–181.
32. Wang, L.; Waltari, O.; Kangasharju, J. MobiCCN: Mobility support with greedy routing in Content-Centric Networks. In Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM), Atlanta, GA, USA, 9–13 December 2013; pp. 2069–2075.
33. Lehmann, M.B.; Barcellos, M.P.; Mauthe, A. Providing producer mobility support in NDN through proactive data replication. In Proceedings of the NOMS 2016—2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, 25–29 April 2016; pp. 383–391.

34. Almeida, V.; Bestavros, A.; Crovella, M.; De Oliveira, A. Characterizing reference locality in the WWW. In Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems, Miami Beach, FL, USA, 18–20 December 1996; pp. 92–103.
35. Psaras, I.; Clegg, R.G.; Landa, R.; Chai, W.K.; Pavlou, G. Modelling and Evaluation of CCN-caching Trees. In Proceedings of the 10th International IFIP TC 6 Conference on Networking (NETWORKING'11), Valencia, Spain, 9–13 May 2011; Part I, pp. 78–91.