A Taxonomy of Caching Approaches in Information-Centric Network Architectures

Andriana Ioannou*, Stefan Weber

School of Computer Science and Statistics, Trinity College, Dublin 2, Ireland

Abstract

The communication paradigm currently used over the Internet resembles that of a telephone communication system where two participants establish an end-to-end connection to exchange information. This model of interaction is being challenged by today's use of the network which focuses on information distribution and retrieval.

Information-Centric Networking (ICN) is an architectural approach, that provides an alternative to this model of interaction by focusing on content distribution, using the network as an intermediate storage. In order to support this, ICN combines a number of concepts such as naming, caching and the publish-subsribe paradigm. The proposed approaches in this area have incorporated a variety of combinations of instances of these concepts and laid out a large solution space including a multitude of possible solutions.

This paper focuses on the issue of caching in ICN and provides a review and a taxonomy of existing approaches based on a number of criteria such as the relation of each approach to forwarding paths, the architectural levels of operation and the awareness of the system regarding the content that is being cached.We further discuss the advantages and the disadvantages of each approach regarding scalability and efficiency and highlight open questions.

Keywords: Network Distributed Architectures; Future Internet; Information-Centric Networks; Caching technologies; Content replication; On-path caching

1. Introduction

The original design of protocols for the Internet provided for information exchange between two participants, following the end-to-end communication model. However, both demand and technologies, have changed since this approach was introduced. Internet usage today is dominated by content distribution and retrieval [1, 2], e.g. Content Delivery Networks (CDNs) for content replication [3–6], Peer-to-Peer (P2P) networks for file sharing [5, 7, 8] as well as media aggregators and social networks [9–11]. This difference between the original communication model and the current usage results in a number of disadvantages and difficulties with regards to availability, mobility, multihoming, scalability and performance [12–16].

CDNs and P2P networks are considered the first steps towards a content-oriented approach. However, these solutions operate at the application layer [5, 6, 8] and clash with underlying assumption of the traditional Internet architecture of the transfer of individual bits of information between two endpoints. This mismatch of higher-layer abstractions and lower-layer implementations may result in reduced performance [17, 18].

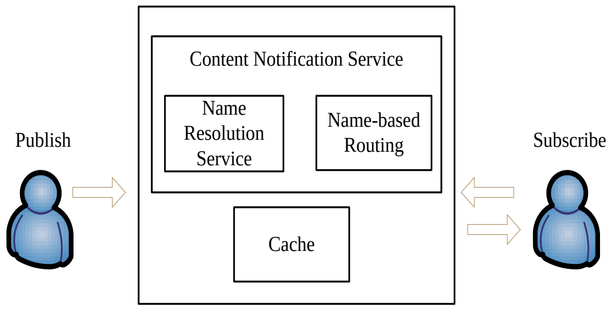

Figure 1: A basic Publish-Subscribe model.

2. The Information-Centric Approach

The *Information-Centric Networking (ICN)* approach is a new architectural model that attempts to provide an alternative to the traditional architecture of the Internet by focusing on information dissemination and information retrieval. The ICN approach has gained the interest of both the research and the industry community, counting a number of projects and individual works, e.g. CCN [13], COMET [19], CONET [17], DONA [16], MultiCache [20], PSIRP-PURSUIT [21, 22], NetInf-SAIL [23] and TRIAD [24].

Proposed ICN architectures differ in a number of aspects, e.g. information granularity, information dissemination techniques or retrieval techniques. However, all of them are based on the basic principles of a publish-subscribe model [25–27]. Figure 1 presents the basic elements and operations of a network following a publish-subscribe model where content sources make their content available by publishing it to a content notification service while clients request content from the content notification service by subscribing to it. Content publication and retrieval is accomplished by identifying the content via names. The publish-subscribe paradigm ensures time, space and syncronization decoupling between publishers and subscribers, facilitating mobility [13, 27].

A content notification service is responsible for matching content publications to corresponding subscriptions. Examples of a content notification service are a *Name Resolution service (NR)* and a *Name-based Routing service* [16, 26]. These approaches may be applied separately or as a combination [17, 26, 28]. A content notification service can be applied in one or more layers of an ICN architecture. Based on this criterion, ICN architectures can be categorized as *"shim"-layer architectures, clean-slate architectures* or *hybrid architectures.*

2.1. Shim-layer Architectures

In a shim-layer architecture, an additional layer acting as an IP overlay, called content layer, is inserted between the transport layer and the network layer of the Open Systems Inteconnection (OSI) stack, [12, 14–17, 19, 24, 26, 28–33]. In this approach, the network layer preserves its current functionality and only the shim layer handles named content. Shim-layer architectures may support both categories of a content notification service, a name resolution service and a name-based routing service. Figure 2 presents the operation of an overlay architecture consists of a source, a client, the shim layer and the current IP network layer components.

2.2. Clean-slate Architectures

In a clean-slate architecture, current network-layer fuctionality of the OSI stack is replaced by content-layer functionality. According to this approach, the network layer is able to handle named contents and no overlay is necessary [13, 17, 21, 22, 34–37]. In contrast to the shim-layer architectures, clean-slate

*Corresponding author. Tel.: +353852742974
E-mail addresses: ioannoa@scss.tcd.ie (A. Ioannou)

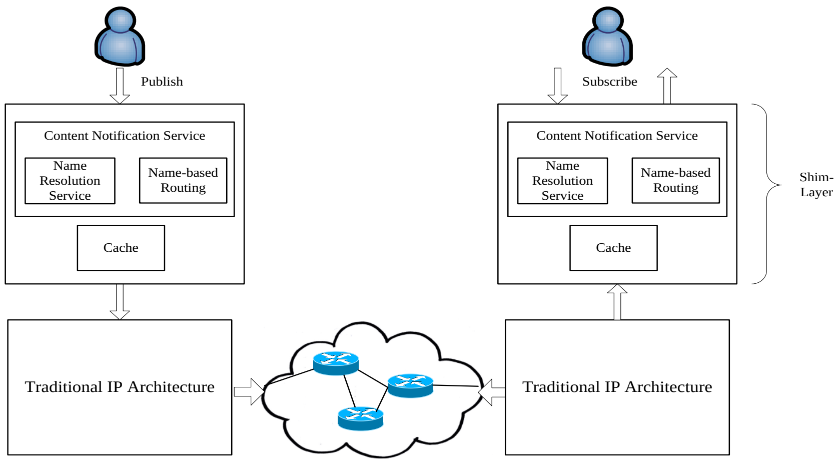

Figure 2: A Shim-Layer ICN architecture.

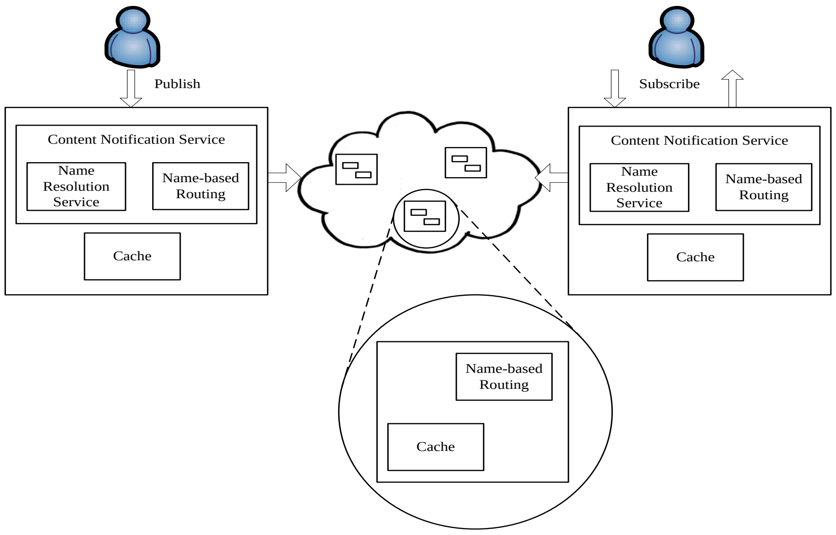

Figure 3: A Clean-Slate ICN architecture.

architectures may only support a name-based routing service and omit any name resolution services. Figure 3 presents the operation of a clean-slate architecture consists of a source, a client and the content-based network layer components.

2.3. Hybrid Architectures

Some ICN architectures may include more than one content layer, i.e. both a shim-layer and a content-based network layer. We call these architectures hybrids due to the fact that they can support both [37]. A hybrid architecture may be represented by combining the Figure 2 and Figure 3, consists of a source, a client, the shim layer and the content-based network layer components.

In general terms, when a content request is released in the network, the content notification service is responsible for identifying the "best" replica, based on the proximity metric(s) used, e.g. number of hops or latency, of that content and deliver it to the requestor. Therefore, ICN can be considered incorporating the anycast principle. In contrast to the CDN and P2P ad-hoc mechanisms that operate at the application layer, ICN mechanisms are rather integrated in the architecture itself.

3. ICN Naming

An important component of each ICN architecture is the definition of naming. Naming can be considered a key for solving the problems that arise from the current Internet architecture, e.g. content being tied to hosts or IP addresses. Ideally, content naming should not involve any topological information that would bind the content to a specific location [12, 16, 19, 23, 26, 38–40]. If this constraint is met, content can be freely replicated and cached in different places and therefore provided by more than one source.

This design of naming functionality strongly influences the scalability and security of an approach and may enable functionality such as authentication and replication. In the design of content namespaces, a number of decisions need to be made such as naming may reveal structural information vs. naming does not reveal structural information, flat naming vs. hierarchical naming, human-readable naming vs. not human-readable naming and self-certified vs. not self-cetrified naming.

Names are used to identify content resources, such as services, web pages, files, songs, videos, etc; but they may also refer to parts of a content resource e.g. chunks or smaller pieces of content such as packets. Chunks describe parts of a content resource as defined by an application e.g. parts 1 to 7 of a movie; whereas packets indicate a division of a content resource that is defined by an underlying network architecture. The content to which a name refers to defines the name granularity of the corresponding architecture. Existing ICN approaches exhibit a variety of levels of naming granularity as shown in Table 1. Currently, three levels of naming granularity can be identified i.e. objects, chunks or packets. It is important to note that ICN approaches may support more than one level of granularity, e.g. the SAIL project supports both object and chunk naming.

Proposal	Objects	Chunks	Packets
CCN [13]			x
COMET [41]	x		
CONET [17]		x	
DONA [16]	x		
MultiCache [20]		x	
PURSUIT [42]	x		
SAIL [43]	x	x	
TRIAD [24]	x		

Table 1: Name granularity in ICN approaches.

4. ICN Content Caching

Caching is considered to be a basic architectural component of an ICN architecture. It may be used to provide a Quality-of-Service (QoS) experience to users, reduce the overall network traffic, prevent network congestion and Denial-of-Service (DoS) attacks and increase availability. Approaches to caching can be categorized into *off-path caching* and *on-path caching* based on the location of caches in relation to the forwarding path from a source to a consumer [26, 42]. Off-path caching, also referred as content replication or content storing, aims to replicate content within a network in order to increase availability, regardless of the relationship of the location to the forwarding path. The actual number of replicas and the specific nodes in which replicas may be stored is a decision made by the Internet Service Provider (ISP) that supports the specific network. In on-path caching approaches, content is replicated at nodes along the forwarding paths from sources to consumers. The decision to cache a content resource at a specific node is strictly related to the content that is being requested. A taxonomy of the proposed ICN caching mechanisms is provided in Figure 4 and Figure 5.

Caching in ICN architectures can be also divided depending on if cached content is propagated into a content notification service. We refer to a system that propagates the existence of the cached content as a *caching-aware system* and to a system that does not propagate the existence of the cached content as

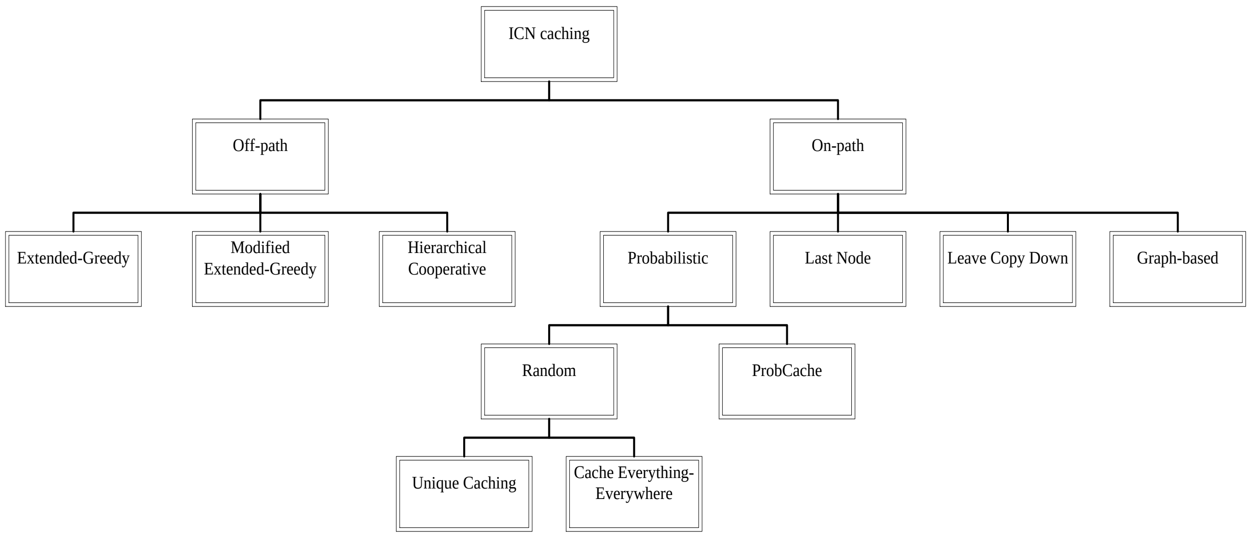

Figure 4: Taxonomy of the ICN caching mechanisms.

an *opportunistic-caching system.* Caching awareness is usually preffered at the off-path caching approach. However, a caching-aware system has also been proposed for the on-path caching approach, e.g. MultiCache [20].

5. Off-path Caching

The optimal placement of content replicas in a distributed storage system has been proven to be a hard task to achieve [44–48]. This conclusion follows from the fact that storage placement decisions may be affected by contextual information, such as node availability, storage availability and the popularity of individual items. In order to provide this information, network monitoring is essential for collecting statistical metrics. The monitoring and collection of contextual information adds to the overhead of the caching mechanism. In addition to this, replicas are usually advertised into a name resolution or a name-based routing service. The distribution of replicas and the update of name resolution systems and name-based routing systems that reference these replicas introduce an additional amount of overhead that needs to be taken into account when deciding about the placement of replicas. The balance of this large set of variables results in a complex system, sensitive to short-term changes which have the potential to increase the overhead of the caching mechanism dramatically. As a result, content replication should be considered as a long term decision [26, 42].

Three approaches, the *Extended-Greedy* algorithm [42], the *Modified Extended-Greedy* algorithm [49] and the *Hierarchical Co-operative* storage system [43], have been proposed as a solution to the placement problem in the ICN literature. The *Extended-Greedy* algorithm is based on the Greedy algorithm [47]. The *Modified Extended-Greedy* algorithm is a modification of the *Extended-Greedy* algorithm based on the assumption that no content origin server exists.

The reason why most ICN proposals have not considered the content placement problem is that content replication in ICN is similar to the CDN content replication and the web cache placement problems [26, 42, 49]. As such, existing algorithms can be used for the ICN off-path caching. A few noticeable examples can be found at [45, 47, 50–52] .

In the following section, we provide a description of the proposed off-path algorithms recorded in the ICN literature. In order for the reader to be able to understand the operation of the Extended-Greedy and the Modified Extended-Greedy algorithms proposed in [42] and [49], we first provide a description of the Greedy algorithm.

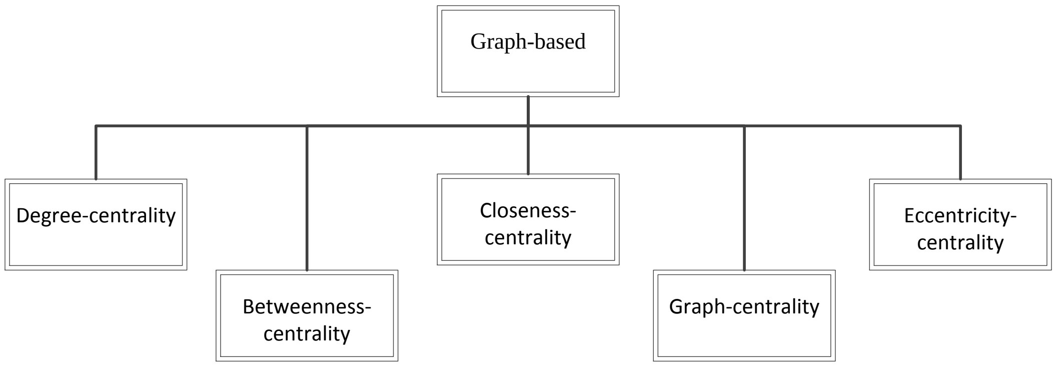

Figure 5: Taxonomy of the Graph-based on-path ICN caching mechanisms.

5.1. *Greedy Placement*

Greedy placement algorithms consitute a family of algorithms that include the *Single-Greedy* [51], *l-Greedy* [53] and *Global-Greedy* placement algorithms [47, 51] as well as the *Greedy-Dual* replacement algorithm [44, 54]. Greedy placement algorithms run for a number of times, at each of which, the node with the lowest cost metric, e.g. latency, hop counts or an economic cost is chosen while previous choices are excluded. After the algorithm has terminated, a set of k nodes out of N available have been selected, where k represents the number of replicas to be made. Qui et al. [47] have shown that a metric cost based on hop counts and request load provides a near to optimal placement solution.

5.2. *Extended-Greedy Placement*

Trossen et al. [42] extended the operation of the Greedy algorithm to be applied to every future replicated item and to take into account popularity rates. In the following, we will describe the operation of the Extended-Greedy algorithm on an example of a graph with six nodes, *N=6*, i.e. $N_1, N_2, N_3, N_4, N_5, N_6$ with a storage capacity of two items per node, $C = 2$, presented in Figure 6. We also assume a source of three items, *t=3*, i.e. t_1, t_2, t_3. The aim is to run the Extended-Greedy algorithm in order to decide the optimal positions for creating two copies of each item, *k=2*. In contrast to the Greedy algorithm, M in the Extended Greedy algorithm represents the number of nodes that need to be chosen for content replication. This number is calculated based on the total number of copies that need to be made, to the number of items that each storage node is able to host, plus one, i.e. $M = (k \times t/C) + 1 = (2 \times 3/2) + 1 = 3 + 1 = 4$.

1. The Greedy algorithm is applied to every future replicated object, called t_i. Once the algorithm has terminated, a vector S_{t_i}, containing the potential k storage nodes, will have been constructed. As request metrics are necessary for this step, we chose a set of nodes to constuct the input vectors randomly.
   Table 2 presents the S_{t_i} vectors corresponding to each individual object as well as the objects' popularity, which is set as a number between *1* and *10*. Note that nodes that have not been selected as potential storage points for an item, e.g. N_6, have been marked using the *0* value. As an example, we explain the table S_{t_1} constructed for item t_1. Assuming that the indexes of the nodes that have been selected as the preferred storage nodes, after the termination of the Greedy algorithm, equals to *1* and the indexes of the ones that have not been selected equals to *0*, vector S_{t_1} will be equal to $S_{t_1} = (0, 0, 1, 1, 0, 0)$.
2. Each vector S_{t_i} is multiplied by a weight w_{t_i} that represents the item's popularity, based on the traffic demand for that item. A new vector, named S, is then calculated by summing the traffic rates for each node. Table 3 presents the traffic rates regarding the popularity of each item as well as the summation of it at each node, *S=(0,7,12,8,2,0)*.
3. We select a set of M nodes that are required for storage, which are the nodes that appear to have the highest sum in vector S. We call this vector S'. According to M, we need to choose four nodes of vector S that hold the highest summation rates. As such, vector *S'* would be $S' = (N_3, N_4, N_2, N_5)$.

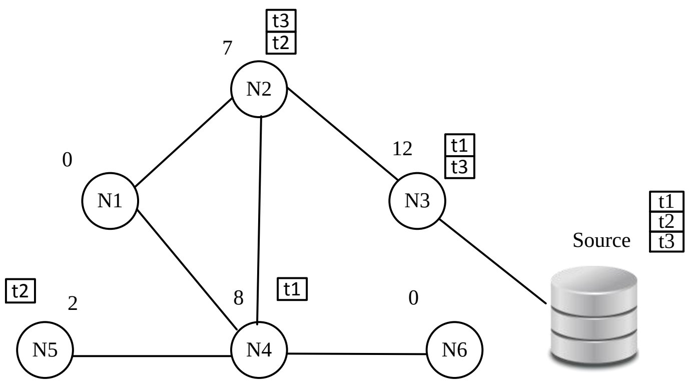

Figure 6: Extended-Greedy placement algorithm.

4. Starting the loop for the most popular item, choosing the first entry of vector S_{t_i} determined on step 1, we place a replica in this node, if and only if, the node is also presented in the S' vector of the final chosen storage nodes and if that node has still storage space available for storing the copy.
   We start illustrating step 4 for each item by following their order of popularity, i.e. t_1, t_3, t_2. Vector S_{t_i} for item t_1 is S_{t_1}= (N_3, N_4). These nodes fulfill the prerequisities of step 4, i.e. they appear at vector S' and there have also enough storage space available for storing the copies. Therefore, N_3 and N_4 are chosen to host item's 1 replicas. The same procedure is executed for each item left. Figure 6 presents the cached copies of each item at each node after the completion of the algorithm.

Item t_i	Vector S_{t_i}	Popularity w_{t_i}
Item t_1	N_3 N_4	8
Item t_2	N_2 N_5	2
Item t_3	N_2 N_3	5

Table 2: Calculation of the S_{t_i} vectors and popularity of each item for the Extended-Greedy algorithm.

	Nodes					
Item $t_i \times w_{t_i}$	N_1	N_2	N_3	N_4	N_5	N_6
Item $t_1 \times w_{t_1}$	0	0	1×8	1×8	0	0
Item $t_2 \times w_{t_2}$	0	1×2	0	0	1×2	0
Item $t_3 \times w_{t_3}$	0	1×5	1×5	0	0	0
Summation S	0	7	12	8	2	0

Table 3: Calculation of the $S_{t_i} \times w_{t_i}$ and S' vectors for the Extended-Greedy algorithm.

While algorithms of the Greedy family may be suitable for replica placement in certain scenarios [47, 53], their suitability in an ICN architecture is still under question. Greedy algorithms are calculated for each item, which means that the cost of the algorithm is increased linearly with the number of replicated items and the size of the network. This factor may be a significant limitation concerning the scalability of the Greedy algorithms.

5.3. Modified Extended-Greedy Placement

The Extended-Greedy placement algorithm has been proposed based on the assumption of existence of a content origin server. However, this assumption is not valid in a publish-subscribe network where

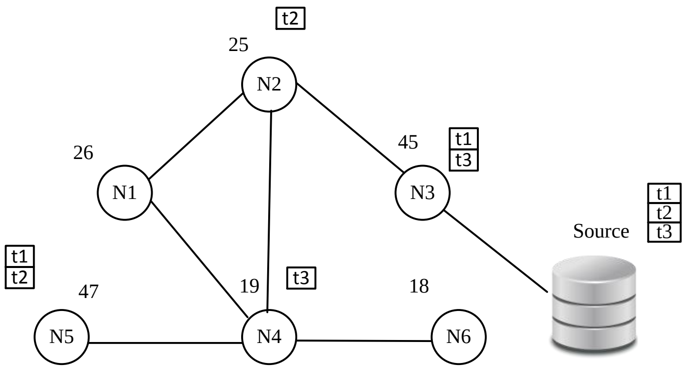

Figure 7: Modified Extended-Greedy placement algorithm.

content may be also published by dynamically joined nodes, acting as clients. In order to overcome the calculation difficulties deriving from the fact that no specific point of reference exists, a modification of the Extended-Greedy algorithm, to which we refer as the Modified Extended-Greedy placement algorithm has been proposed [49].

The operation of the Modified Extended-Greedy placement algorithm is quite similar to the one of the Extended-Greedy placement algorithm described in section 5.2. In this section we use the same example presented in section 5.2, Figure 7, to illustrate the operation of the Modified Extended-Greedy algorithm and to point out the differences between the two.

1. The Modified Extended-Greedy algorithm repeats the operation of the Greedy algorithm N times for each item t_i, where N is the number of nodes in the network, based on the following assumptions:
   - Only k nodes out of N are selected each time, where k is the number of replicas that are required for each content item.
   - The origin server is represented by a different node each time. As such, a set of N vectors, presenting the selected storage nodes, will have been constructed after the execution of the algorithm. We name these vectors S_{t_i,N_i}, where t_i presents the item for which the algorithm has been executed and i presents the origin server node. As request metrics are neccessary for this step, we rather chose a set of nodes to construct our tables randomly. Table 4 summarizes the calculations of the S_{t_i,N_i} vectors of each item t_i after the execution of the Modified Extended-Greedy algorithm. Similarly to the assumptions used in section 5.2, where the index of a chosen storage node equals to *1* and the index of a non chosen storage node equals to *0*, vector S_{t_1,N_1} would be equal to S_{t_1,N_1}*=(0,1,1,0,0,0)*.
   - A vector named S_{t_i} is calculated for each item t_i. Vector S_{t_i} presents the summation of the times that each node N_i has been recorded concerning each item t_i, after the N times of execution of the algorithm, i.e. S_{t_i} vectors are calculated by summing the individual S_{t_i,N_i} vectors. Accordingly, vectors S_{t_1}, S_{t_2} and S_{t_3} of items t_1, t_2 and t_3 will be: $S_{t_1} = (2,2,3,0,3,2), S_{t_2} = (0,2,3,2,4,1)$ and $S_{t_3} = (2,1,3,3,3,0)$, respectively. As an example, $S_{t_1} = (2,2,3,0,3,2)$ means that out of the *N=6* times of execution of the Greedy algorithm, nodes N_1, N_2 and N_6 have been recorded two times each, nodes N_3 and N_5 have been recorded three times each while node N_4 has been recorded zero times.
2. In a similar way as in step 2 of the Extended-Greedy algorithm, each vector S_{t_i} is multiplied by a factor w_{t_i}, the popularity of each item, while a new vector, named S, is calculated, consists of the traffic rates of each node N_i. Table 5 presents the calculation of the S_{t_i} vectors multiplied by w_{t_i} as well as the summation of these values, i.e. the calculation of vector *S=(26,25,45,19,47,18)*.
3. Starting from the nodes that hold the highest values in vector S, we select M nodes out of the N that are required for the storage of the replicas. We call this vector *S'*. Following the same example as in section 5.2, we have that *M=4*. Therefore, vector *S'=*(N_5, N_3, N_1, N_2).

4. Starting from the most popular item, i.e. t_1, t_3, t_2, we store a replica of item t_i in the k most appeared nodes presented in vector S_{t_i}, if and only if, they are included in vector *S'* and there is still enough space available. In the case that not enough space is available the algorithm proceeds into checking the next node of vector S_{t_i} based on the rates of appearance. As an example, we illustrating step 4 for the item t_1 with a vector $S_{t_1} = (N_1, N_2, N_3, N_5, N_6)$ and values of it $S_{t_1} = (2, 2, 3, 0, 3, 2)$. As variable k indicates we need to choose two nodes. In contrast to the Extended-Greedy algorithm where no preference is given, in the Modified Extended-Greedy algorithm we choose the nodes that have the highest appearance rates. In the case that more than one have the same value, a random decision is made. Therefore, we choose the nodes N_3 and N_5. Both nodes fulfill both requirements, i.e. they are included in vector *S'* and have available storage place. As such, nodes N_3 and N_5 are chosen to store item's 1 replicas. The replication nodes for the rest of the items t_3, t_2 can be calculated by following the same procedure. Figure 7 presents the final items stored after the completion of the Modified Extended-Greedy algorithm.

	Origin Server					
Item t_i	N_1	N_2	N_3	N_4	N_5	N_6
Item t_1	N_2, N_3	N_3, N_5	N_5, N_6	N_1, N_6	N_1, N_2	N_3, N_5
Item t_2	N_3, N_5	N_2, N_5	N_4, N_6	N_3, N_5	N_2, N_4	N_3, N_5
Item t_3	N_3, N_5	N_1, N_4	N_2, N_5	N_3, N_4	N_1, N_4	N_3, N_5

Table 4: Calculation of the S_{t_i,N_i} vectors for the Modified Extended-Greedy algorithm.

	Nodes					
Item $t_i \times w_{t_i}$	N_1	N_2	N_3	N_4	N_5	N_6
Item $t_1 \times w_{t_1}$	2×8	2×8	3×8	0	3×8	2×8
Item $t_2 \times w_{t_2}$	0	2×2	3×2	2×2	4×2	1×2
Item $t_3 \times w_{t_3}$	2×5	1×5	3×5	3×5	3×5	0
Summation S	26	25	45	19	47	18

Table 5: Calculation of the $S_{t_i} \times w_{t_i}$ vectors and the S vector for the Modified Extended-Greedy algorithm.

As part of the Greedy algorithms family, the Modified Extended-Greedy algorithm holds all the disadvanges that derive from the nature of a greedy algorithm. In addition to this, Modified Extended-Greedy algorithm increases the calculation time that is necessary to be accomplished due to the fact that no origin server exists.

5.4. Hierarchical Co-operative placement

Ericsson et al. [43] have suggested the operation of a hierarchical co-operative placement system. Co-operative caching, however, introduces a significant amount of overhead to the network due to the signaling that is required between the co-operative caching nodes. The first attempt in solving the co-operative placement problem is the exploitation of the Name Resolution System (NRS) that their architectural model provides. According to this approach, the NRS is exploited for quering the location of other replicas available as well as the popularity of each of them based on the request rates observed by the NRS; the NRS operates as a "common" information base, reducing the information that co-operative nodes need to exchange with each other.

An alternative to this approach, exploits the Multiple Distributed Hash-Table (MDHT) location system in the architectural model of their system [55]. MDHT utilises three different levels in the NRS, i.e. the access-node level, the Point-of-Presence (POP) level and the Autonomous-System (AS) level. The basic assumption of that approach is that instead of storing at every level, storing is available only in a specific level(s).

Based on that distinction, content placement can be divided into two different approaches, the one-level content placement approach and the multi-level content placement approach. In the former caching is carried out in only one level of the architecture, e.g. the node-level, the POP-level or the AS-level, while in the latter items can be stored at multiple levels according to their popularity, e.g. POP/AS, where the most frequently requested objects are stored at the POP-level while the next most frequently requested objects are stored at the AS-level. A key question relating to this caching approach is exactly which level(s) provide(s) the greatest benefits. Towards trying to answer this question, Ericsson et al. [43] tested all the above mentioned approaches in terms of hit rates and latency, concluding into three basic observations:

1. AS-level is the most beneficial approach.
2. POP/AS multi-level approach seems to follow the AS-level storing approach.
3. Node-level storing is considered to perform the worst.

According to the AS-level co-operative storing approach, each AS-node is able to store content depending on the content's popularity, yet all of them should co-operate in order to prevent content redundancy. The goal is to keep the most popular contents cached by evicting the least popular ones. They name the selected replacement algorithm *Forward MetaData (FMD).*

We argue that Hierarchical co-operative off-path caching proposal may not be beneficial due to the amount of traffic that the collaborative caching model is introducing as well as the computational resources that are neccessary in order for this communication to be achieved [54, 56, 57]. In addition to this, co-operative caching benefits are under question as measurements on real large web traces have shown that in large organizations or large populations, it is rather unlikely to have significant benefits [58]. Furthermore, the proposed approach lacks of important details regarding basic features, e.g. how the collaborative caching is accomplished while the approach has not been tested against other approaches.

6. On-path Caching

On-path caching has been suggested as an integrated ICN architectural mechanism. In contrast to off-path caching, on-path caching decisions applies only to the requested content(s); other content is not taken into account, while content may be cached only at the nodes lying on the delivery path. On-path caching is strictly related to the requested content and popularity rates of each item. As a result, on-path caching may be considered as a short-term procedure.

On-path caching is also related to the content's name granularity. Based on this criterion, on-path caching can be divided into three categories, i.e. *object-level caching, chunk-level caching* and *packet-level caching.* A combination of more than one approaches can also be applied, e.g. on-path object-level and on-path packet-level caching [59].

- Object-level on-path caching

  In the object-level approach, full objects can be cached at the intermediary nodes along the delivery path. Object-level on-path caching is typically performed at the overlay level of an ICN architecture.

- Chunk-level on-path caching

  Chunk-level on-path caching is accomplished in the same way as the object-level version. The only difference is in the granularity of the name. As in object-level caching, chunk-level caching is also accomplished at the overlay level of an ICN architecture.

- Packet-level on-path caching

  Packet-level on-path caching is performed by naming each packet individually. The naming of each packet should be, somehow, related to the object that it corresponds to. Packet-level on-path caching is performed at the networking layer of an ICN architecture.

The level based on which on-path caching may be applied is a critical design issue, having an overall effect at the system's performance and operation. We briefly examine two examples, the memory and the computational requirements that an on-path caching node should fulfill. Memory requirements are expected to be much higher for the object-level on-path caching approach while computational requirements are expected to be much higher for the packet-level approach. Our conclusion is based on the assumption that object-level caching requires more memory for caching just one content compared to the packet-level approach. In the same sense, computational requirements for a lookup would be higher for the packet-level approach compared to the object-level one, as more contents can be cached for the same amount of memory.

On-path caching schemes can be also divided based on the caching technique that is used. A few attempts on identifying such algorithms have been recorded in the literature as part of known projects or individual works. In this section we provide a taxonomy of the proposed approaches and discuss their advantages and disadvantages. Table 9 summarizes the proposed on-path caching approaches as well as the ones that they have been compared to. It also presents their level of operation as well as the nature of the system, i.e. *caching-aware* system or *opportunistic-caching* system. In this table, "-" is used so as to indicate that no further information has been provided regarding that category. At this point it is important to note that the caching approaches proposed in the COMET project represent a rather general piece of work and do not follow the principles defined for the COMET architecture, i.e. caching has been proposed to work at the packet-level while the COMET architecture considers only complete content objects.

Proposal	Proposed Technique	Comparison Technique	Caching-Level	Caching System
Caching performance of content centric networks under multi-path routing (and more) [60]	LCD	CE^2, RND	chunk	opportunistic
On sizing CCN content stores by exploiting topological information [61]	DC, SC, BC, CC, GC, EC	each other	chunk	opportunistic
Cache "Less for More" in Information-Centric Networks [62]	BC	UniCache, CE^2	object, chunk, packet	opportunistic
CCN [13]	CE^2	-	packet	opportunistic
COMET [63, 66]	ProbCache	CE^2	packet	opportunistic
	BC	CE^2	packet	opportunistic
CONET [17]	-	-	chunk	opportunistic
DONA [16]	-	-	object	opportunistic
MultiCache [20]	LastNode	-	chunk	caching-aware
On content-centric router design and implications [64]	RND	CE^2	packet	opportunistic
Probabilistic in-network caching for information-centric networks [65]	ProbCache	CE^2, LCD, RND	chunk	opportunistic
PURSUIT [59]	CE^2	-	object, packet	opportunistic
SAIL [43]	CE^2	-	object, chunk	opportunistic
TRIAD [24]	-	-	object	opportunistic

Table 6: Taxonomy of the proposed on-path caching algorithms.

6.1. Probabilistic Caching

Probabilistic caching is a general approach, according to which each node on the delivery path decides to cache the content based on a probability p [64, 66, 67]. The probability p may be a pre-determined value [64, 68] or may be calculated based on a mathematical formula, composed of a number of individual components [66].

6.1.1. Random Caching

In *random caching (RND)*, probability p is set at a standardized value, e.g. p=1, 1/2, 1/3 etc. [64]. *RND* caching decisions are individually taken, involving no coordination between the nodes. This model is fairly simple and results in no additional load on the network. However, it is not able to exploit the advantage of having knowledge of the optimal positions for caching each content, based on, for example, the content's popularity. In addition to this, random decisions can lead in high content redundancy or in no caching at all.

- Unique Caching

  Unique caching is a form of randomized on-path caching. For the rest of the paper we will refer to this caching approach as the *UniCache* approach. In UniCache, content is cached only in one node along the delivery path which is chosen randomly [62, 68]. Since only node is chosen, the probability of caching at each node equals to, one to the number of intermediary nodes. Considering an example of four intermediary nodes, the caching probability for unique caching would be *p=1/4* at each node. As a subcategory of the random caching approach, unique caching holds all the disadvantages deriving from its parent.

- Caching Everything-Everywhere

  Jacobson et al. [13] have proposed an approach called *Caching everything-everywhere* (CE^2) as the preferred on-path caching approach. The CE^2 approach has also been adopted by other ICN proposals [26, 37]. The CE^2approach simply caches every content in every intermediate node involved in the delivery path. The CE^2 approach has been criticized in a number of works [54, 60, 62, 64, 66, 67, 69–72] for resulting into uneccessary content redundancy and resource consumption. As an additional drawback, CE^2 does not take into account the content's popularity, providing the same probability, for both popular and unpopular content, to be cached. In contrast to its disadvantages, CE^2 holds the advantage of providing fast content distribution [62].

6.1.2. ProbCache

Kamel et al. [66] suggested a probabilistic caching algorithm that they name, *ProbCache. ProbCache* is assumed to work at a packet-level or at a chunk-level [65]. *ProbCache* is based on two factors, the *TimesIn* factor and the *CacheWeight* factor. *ProbCache* is calculated at each node lying on the delivery path.

$$ProbCache = \underbrace{\frac{\sum_{i=1}^{x-(y-1)} N_i}{R_c}}_{TimesIn} \times \underbrace{\frac{y}{x}}_{CacheWeight} \tag{1}$$

$$ProbCache = \underbrace{\frac{\sum_{i=1}^{x-(y-1)} N_i}{T_{tw} \times N_x}}_{TimesIn} \times \underbrace{\frac{y}{x}}_{CacheWeight} \tag{2}$$

The goal of the algorithm is to provide fairness regarding the available capacity of the delivery path. In order to achieve that, values x and y correspond to the number of hops travelled from the requestor to the content source and to the number of hops travelled from the source towards to the requestor, respectively. The x value remains stable during the delivery of the content. As such, ProbCache is calculated based on the capacity of the remaining nodes left at the delivery path. According to the authors, *TimesIn* factor favors contents that travel from further away while *CacheWeight* factor acts as a counter-balance to this unfairness.

The *TimesIn* factor indicates the number of replicas of the content that are expected to be cached along the delivery path. The *TimesIn* factor depends on the R_c value, equation 1, which ideally indicates the number of different content items that a given delivery path has to cache. However, in a real-time network, this value cannot be calculated. Therefore, R_c could be considered as the number of the content items that each node has to process per time unit. In a newest version of the algorithm, provided by the same authors

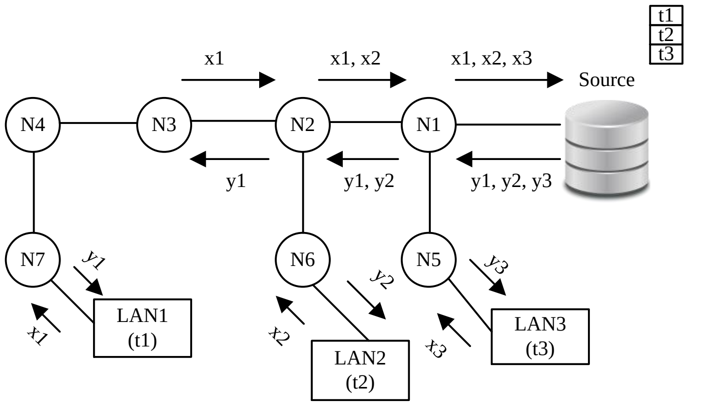

Figure 8: ProbCache caching algorithm.

in [65], R_c value is replaced by the combination of the T_{tw} and N_x variables, equation 2, where T_{tw} value is the time for which any content should be kept by any cache lying on the delivery path and N_x value is the average cache size along the same path. The T_{tw} value depends on the size of the caches. Assuming that nodes have a capacity of 10GBytes DRAM [64], a value of 10 seconds is determined.

In order to evaluate the suitability of both the proposed ProbCache algorithms, equations 1 and 2, we illustrate an example. In our example, presented in Figure 8, we assume a network of seven nodes, N_1, N_2, N_3, N_4, N_5, N_6, N_7, with each one having the same amount of cache memory. We also assume a source of three objects, t_1, t_2 and t_3 and three clients, *Client1, Client2* and *Client3*, sending content requests for the objects t_1, t_2 and t_3, respectively. In order to make the figure as comprehensible as possible, we have added arrows showing the direction of the content requests, i.e. x_1, x_2, x_3 and the delivery of them y_1, y_2, y_3. Table 7 presents the values of x and y for each item at each individual node lying on the delivery path as well as the R_c values. The size of the cache depends on the assumptions of each ProbCache algorithm that has been proposed. Regarding the first proposed ProbCache algorithm, equation 1, R_c value corresponds to the number of items that each node has to deliver at each time. As such, cache size should be similarly calculated. We do assume that each node's capacity equals to the size of two content objects. Regarding the second proposed ProbCache algorithm, equation 2, storage capacity of each node is assumed to be 10GBytes. As such, T_{tw} equals to 10. Similarly, since all nodes have the same amount of cache memory, N_x equals to 10 as well. In order to evaluate the suitability of both the proposed ProbCache algorithms, equations 1

	Nodes						
	N_1	N_2	N_3	N_4	N_5	N_6	N_7
	x_1=5	x_1=5	x_1=5	x_1=5	-	-	x_1=5
	x_2=3	x_2=3	-	-	-	x_2=3	-
	x_3=2	-	-	-	x_3=2	-	-
	y_1=1	y_1=2	y_1=3	y_1=4	-	-	y_1=5
	y_2=1	y_2=2	-	-	-	y_2=3	-
	y_3=1	-	-	-	y_3=2	-	-
R_c	3	2	1	1	1	1	1

Table 7: Values of x and y at each node during delivery process.

and 2, we calculate the ProbCache values at node N_2 for the items t_1 and t_2, i.e. equations 3, 4, 5 and 6. Equations 3 and 4 illustrate the operation of equation 1, while equations 5 and 6 illustrate the operation of equation 2, respectively. ProbCache values for the rest of the nodes may be calculated in a similar manner.

$$ProbCache(t_1) = \underbrace{\frac{\sum_{i=1}^{x_1-(y_1-1)} N_i}{R_c}}_{TimesIn} \times \underbrace{\frac{y_1}{x_1}}_{CacheWeight} = \underbrace{\frac{\sum_{i=1}^{5-(2-1)} 2}{2}}_{TimesIn} \times \underbrace{\frac{2}{5}}_{CacheWeight} = \frac{8}{5} \quad (3)$$

$$ProbCache(t_2) = \underbrace{\frac{\sum_{i=1}^{x_2-(y_2-1)} N_i}{R_c}}_{TimesIn} \times \underbrace{\frac{y_2}{x_2}}_{CacheWeight} = \underbrace{\frac{\sum_{i=1}^{3-(2-1)} 2}{2}}_{TimesIn} \times \underbrace{\frac{2}{3}}_{CacheWeight} = \frac{4}{3} \quad (4)$$

As indicated from the values calculated at equations 3 and 4, ProbCache values may be higher than the defined range [0,1]. These kind of values are not acceptable. Therefore, it appears that the first proposal of the ProbCache algorithm, as it is identified by equation 1, is not a proper caching management technique.

$$ProbCache(t_1) = \underbrace{\frac{\sum_{i=1}^{x_1-(y_1-1)} N_i}{T_{tw} \times N_x}}_{TimesIn} \times \underbrace{\frac{y_1}{x_1}}_{CacheWeight} = \underbrace{\frac{\sum_{i=1}^{5-(2-1)} 10}{10 \times 10}}_{TimesIn} \times \underbrace{\frac{2}{5}}_{CacheWeight} = \frac{8}{50} \quad (5)$$

$$ProbCache(t_2) = \underbrace{\frac{\sum_{i=1}^{x_2-(y_2-1)} N_i}{T_{tw} \times N_x}}_{TimesIn} \times \underbrace{\frac{y_2}{x_2}}_{CacheWeight} = \underbrace{\frac{\sum_{i=1}^{3-(2-1)} 10}{10 \times 10}}_{TimesIn} \times \underbrace{\frac{2}{3}}_{CacheWeight} = \frac{4}{30} \quad (6)$$

Eventhough equations 5 and 6 correspond to acceptable probabilistic values that fall into the [0,1] range, we are concerned that *ProbCache* is not a beneficial approach due to the *TimesIn* factor, due to the T_{tw} and N_x values which remain stable for every item at any node. In addition to this, the algorithm does not consider the popularity of the content while it requires that probability should be calculated at each node involved in the delivery path. This approach could introduce high computational cost, compared at least to simpler ones, such as random caching.

6.2. Last-Node Caching

Last-Node caching has been proposed as an on-path caching approach operating at the overlay layer of the proposed *MultiCache* ICN architecture [20]. We will refer to this approach as the *LastNode* caching approach. According to this approach, content is cached at the last node participating at the overlay architecture of the delivery path, from the source towards to the requestor, while is also advertised at the ancestors of the last node, upwards to the source. In contrast to other proposals where only the node to which a request arrives is being checked, downstream nodes are also checked in a *Depth-First Search (DFS)* fashion, in case that the node has transmitted the requested object to its descendants in a previous time. If multiple child nodes have cached the potential content, preference is given to the one that is at the same AS with the overlay node initiating the request. In the case where more than one exist, a random decision is made.

We argue that LastNode caching is a non-beneficial approach due to its additional traffic of advertising the cached content to its predecessors. In addition to this, LastNode caching exploits only a specific group of nodes while the approach has not been tested against other approaches.

6.3. Leave Copy Down Caching

Leave Copy Down (LCD) [67] caching was initialy proposed for the multilevel web caching problem, as an alternative to the de facto *Leave Copy Everywhere (LCE)* caching approach or CE^2, as we refer to it at this paper. However, the *LCD* caching algorithm has recently been proposed and tested against other proposed ICN caching algorithms [60]. The *LCD* algorithm simply caches a copy of the requested content one node closer to the client each time that a content request arrives. The *LCD* caching approach differs from the CE^2 approach in the sense that it only copies the object further when an additional content request occurs while CE^2 caches a copy in all the nodes of the delivery path upon the arrival of the very first content request.

We do consider that *LCD* might be a rather good approach for the on-path caching placement problem as it does not introduce any additional computational costs, yet it takes into account the object's popularity, which is indicated by the number of requests concerning that object. Recent metrics based on the LCD algorithm verify our claims [60, 68].

6.4. Graph-related Centrality-based Caching

A number of graph-related centrality-based on-path caching algorithms have been proposed for the ICN architectures [61], including *Degree Centrality (DC)*, *Betweenness Centrality (BC)*, *Closeness Centrality (CC)*, *Graph Centrality (GC)* and *Eccentricity Centrality (EC)*. Each of these algorithms is applied at each potential caching node, individually. In this section we provide a brief description of each algorithm. For the rest of the section, we do assume the usage of the shortest path as the routing metric. However, any kind of metric can be used instead.

In order to be able to clearly describe all the graph-related centrality-based algorithms we use the same graph as presented in Figure 6 and Figure 7. Based on that graph, Table 8 presents the intermediary nodes that lie on the calculated shortest paths between two nodes, where "-" indicates that the nodes are directly connected to each other. Providing an example, the shortest path between the nodes N_3 and N_5 is as follows: (N_3, N_2, N_4, N_5). Based on the same graph, Table 9 summarizes all the calculated graph-based values for each node.

	Nodes					
	N_1	N_2	N_3	N_4	N_5	N_6
N_1		-	N_2	-	N_4	N_4
N_2	-		-	-	N_4	N_4
N_3	N_2	-		N_2	N_2, N_4	N_2, N_4
N_4	-	-	N_2		-	-
N_5	N_4	N_4	N_4, N_2	-		N_4
N_6	N_4	N_4	N_4, N_2	-	N_4	

Table 8: Shortest path calculations between two nodes.

6.4.1. Degree Centrality-based Caching

The *Degree Centrality-based (DC)* caching [61] algorithm is based on the number of links that correspond to each node *n*. As an example the *DC* value of node N_4 would be $DC_{N_4} = 4$. Rossi et al. in [61] have concluded that *DC* is the most effective graph-related metric compared to the alternatives.

6.4.2. Betweenness Centrality-based Caching

The *Betweenness centrality-based (BC)* caching algorithm [61, 62, 66] is based on the concept of betweenness centrality, originated from the area of social networks. The *BC* metric represents the number of times that each node *n* lies at the sets of shortest paths, between all the pairs of nodes in the graph, besides *n*. The idea is that if a node lies in many delivery paths, it has a higher probability of experiencing a cache hit.

As an example we calculate the *BC* value of node N_4. In order to do this we need to be aware of all the shortest paths between all the other nodes, i.e. nodes N_1, N_2, N_3, N_5 and N_6. Depending on the calculations provided in Table 8, we count a number of seven shortest paths that pass through node N_4, therefore, BC_{N_4}=7.

6.4.3. Closeness Centrality-based Caching

The *Closeness centrality-based (CC)* algorithm calculates the reverse sum of the shortest path distances between each node n and the rest of the nodes in the graph. The lowest this value is, the more cantralized is the node.

Again, as an example, we calculate the CC value of node N_3. The shortest path distances between node N_3 and each of the other nodes, i.e. N_1, N_2, N_4, N_5, N_6 are *1,0,1,2,2*, respectively. According to these values, the CC value of node N_3 is CC_{N_3}*=1/(1+0+1+2+2)=1/6.*

6.4.4. Graph Centrality-based Caching

The *Graph centrality-based (GC)* value corresponds to the reverse maximum shortest path distance value from node n to all the other nodes in the graph. The idea is that high-GC nodes will act as "central" nodes, handling more content requests, thus, increasing the cache hit rates.

As an example, based on the shortest path distances calculated for node N_3 at Table 8, node N_3 has a maximum distance of 2 towards all the rest nodes of the graph. As such, the GC value of node N_3 is GC_{N_3}*=1/2.*

6.4.5. Eccentricity Centrality-based Caching

The *Eccentricity centrality-based (EC)* value reflects the maximum distance between node n and each other node in the graph. EC equals to the reverse of the GC value, i.e. $EC = 1/GC$. Based on the previous example for node N_3, the EC value of it would be $EC_{N_3} = 2$.

	N_1	N_2	N_3	N_4	N_5	N_6
DC	2	3	1	4	1	1
BC	0	4	0	5	0	0
CC	1/3	1/2	1/6	1	1/5	1/5
GC	1	1	1/2	1	1/2	1/2
EC	1	1	2	1	2	2

Table 9: Calculation of each graph-based value at each node.

Graph-related caching algorithms have been proposed as representative metrics for determing the size of the caches, e.g. proportional to the centrality value of each node rather than determing the nodes for caching the content. However, as indicated by Kamel et al. [66], graph-related centrality-based algorithms may be also used for solving the on-path caching placement problem. Even though the proposed packet-level caching algorithm illustrates the operation of the BC algorithm, any other graph-related centrality-based metric. Accordingly, we provide a generalized description of it.

In such a scenario, a topology manager calculates and assigns the centrality values at each node in an off-line manner. Caching nodes are chosen based on their corresponding centrality value. In more detail, the algorithm assumes that during the request path towards the content source, each node's centrality value is checked. The value is updated at each node and stored in the packet header. Once the request packet reaches the content source it would have the highest or the lowest centrality value recorded. As an example, considering the DC and SC algorithms, the nodes with the highest centrality values will be chosen, while considering the CC algorithm, the nodes with the lowest centrality value will be chosen. This value is then used as a caching criterion, i.e. content is cached only at the nodes involved in the delivery path who have their centrality value equal to the one recorded. A disadvantage deriving from the proposed algorithm is that only a specific set of nodes is chosen, leaving the capacity of all the other nodes participating at the delivery path unexploited.

A general disadvantage deriving from the nature of the graph-related caching algorithms is their limited efficiency. We do argue that the efficiency of a centrality-based algorithm is topologically limited as centralized information is required for its operation, e.g. a topology manager. Therefore, centrality-based algorithms may operate properly only in an AS-scale. The reason for this is that under a realistic scenario

ASes do not exchange any traffic information so as to conclude to a globally unique centrality value. Even under an optimistic scenario, the scalability issues of a global topology manager would be an open question. The *DC* algorithm constitutes the exception to this rule. Although, *DC* approach is able to work fine in a decentralized global network architecture we do argue that a better and more representative metric would be the load of each node, i.e. the number of requests that a node receives regarding a specific period of time.

An important drawback, however, related to all the centrality-based algorithms, is that they do not consider the fact that a node may become overloaded due to its highly traffic position inside the network. Hence, centrality-based algorithms may introduce significant latency and delay. Furthermore, centrality-based algorithms do not take into account the popularity rates of the content.

7. Discussion

Even though, caching has been extensively investigated in many fields, e.g. web proxies and CPUs, ICN caching and on-path caching in particular can be described as a rather new area of research. The main reason for this argument is the integration of the caching mechanisms into the network architecture. As such, caching mechanisms should be "friendly" enough to the rest of the architectural components. As an example, on-path caching operational requirements should introduce as less delay as possible to the system.

As indicated in Table 6, it appears that ICN caching mechanisms have not given sufficient attention in the ICN literature. In addition, each of these algorithms has been proposed as part of individual projects or works. As a result, algorithms may be tied to the architectural characteristics of the different ICN architectures for which they have been proposed, e.g. *LastNode* approach where cached contents are advertised to the predecessors of the cache node.

Furthermore, some of the proposed on-path caching algorithms have been compared only against the CE^2 approach, this applies to the *BC* and *RND* caching algorithms, while others have not been compared against any of the existing ones, including the *LastNode* caching. As such, no conclusion can be made regarding the efficiency of the proposed algorithms and their suitability for an ICN architecture.

8. Conclusions

In this paper we have extensively described the proposed caching algorithms for the ICN architectural model and categorized them against their properties. We have further discussed and analyzed the advantages and disadvantages of each of them. We argue that ICN caching is an important architectural mechanism which has not been satisfactory investigated. Further research is necessary in order to understand the operation and the requirements of an ICN caching system and be able to design more proper algorithms.

9. Acknowledgements

At this section we would like to thank you all the people that helped us into concluding to the statements included in this paper. In particular, we would like to give our special thanks to our collegues Elwyn Davies and Guoxian Yang for their valuable comments.

10. References

[1] D. Reinsel, C. Chute, W. Schlichting, J. McArthur, S. Minton, I. Xheneti, A. Toncheva, A. Manfrediz, The expanding digital universe, White paper, IDC.

[2] C. Index, Forecast and methodology, 2009-2014, White paper, CISCO 2.

[3] R. Buyya, M. Pathan, Content delivery networks, Vol. 9, Springer, 2008.

[4] G. Pallis, A. Vakali, Insight and perspectives for content delivery networks, Communications of the ACM 49 (1) (2006) 101–106.

[5] A. Passarella, A survey on content-centric technologies for the current internet: Cdn and p2p solutions, Computer Communications 35 (1) (2012) 1–32.

[6] A. Vakali, G. Pallis, Content delivery networks: Status and trends, Internet Computing, IEEE 7 (6) (2003) 68–74.

[7] S. Androutsellis-Theotokis, D. Spinellis, A survey of peer-to-peer content distribution technologies, ACM Computing Surveys (CSUR) 36 (4) (2004) 335–371.

[8] E. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, A survey and comparison of peer-to-peer overlay network schemes, IEEE Communications Surveys and Tutorials 7 (2) (2005) 72–93.

[9] A. Kaplan, M. Haenlein, Users of the world, unite! the challenges and opportunities of social media, Business horizons 53 (1) (2010) 59–68.

[10] A. Mislove, H. Koppula, K. Gummadi, P. Druschel, B. Bhattacharjee, Growth of the flickr social network, in: Proceedings of the 1st workshop on Online social networks, ACM, 2008, pp. 25–30.

[11] A. Mislove, M. Marcon, K. Gummadi, P. Druschel, B. Bhattacharjee, Measurement and analysis of online social networks, in: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, ACM, 2007, pp. 29–42.

[12] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, M. Walfish, A layered naming architecture for the internet, in: ACM SIGCOMM Computer Communication Review, Vol. 34, ACM, 2004, pp. 343–352.

[13] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, R. Braynard, Networking named content, in: Proceedings of the 5th international conference on Emerging networking experiments and technologies, ACM, 2009, pp. 1–12.

[14] F. Al-Shraideh, Host identity protocol-extended abstract, in: Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, ICN/ICONS/MCL 2006, IEEE, 2006, p. 203.

[15] A. Jonsson, M. Folke, B. Ahlgren, The split naming/forwarding network architecture, in: First Swedish National Computer Networking Workshop (SNCNW 2003), 2003.

[16] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, I. Stoica, A data-oriented (and beyond) network architecture, in: ACM SIGCOMM Computer Communication Review, Vol. 37, ACM, 2007, pp. 181–192.

[17] A. Detti, N. Blefari Melazzi, S. Salsano, M. Pomposini, Conet: a content centric inter-networking architecture, in: Proceedings of the ACM SIGCOMM workshop on Information-centric networking, ACM, 2011, pp. 50–55.

[18] J. Pan, Y. Hou, B. Li, An overview of dns-based server selections in content distribution networks, Computer Networks 43 (6) (2003) 695–711.

[19] G. Garcia, A. Beben, F. Ramon, A. Maeso, I. Psaras, G. Pavlou, N. Wang, J. Sliwinski, S. Spirou, S. Soursos, et al., Comet: Content mediator architecture for content-aware networks, in: Future Network and Mobile Summit (FutureNetw), 2011, IEEE, 2011, pp. 1–8.

[20] K. Katsaros, G. Xylomenos, G. Polyzos, Multicache: An overlay architecture for information-centric networking, Computer Networks 55 (4) (2011) 936–947.

[21] N. Fotiou, D. Trossen, G. Polyzos, Illustrating a publish-subscribe internet architecture, Telecommunication Systems 51 (2012) 233–245.

[22] N. Fotiou, P. Nikander, D. Trossen, G. Polyzos, Developing information networking further: From psirp to pursuit, Broadband Communications, Networks, and Systems 66 (2012) 1–13.

[23] C. Dannewitz, Netinf: An information-centric design for the future internet, in: Proceedings of the 3rd GI/ITG KuVS Workshop on The Future Internet, 2009.

[24] D. Cheriton, M. Gritter, Triad: A new next-generation internet architecture (2000).

[25] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, V. Vercellone, Design considerations for a network of information, in: Proceedings of the 2008 ACM CoNEXT Conference, no. 66 in CoNEXT '08, ACM, 2008, pp. 1–6.

[26] A. Ericsson, The network of information: Architecture and applications, Tech. rep., FP7-ICT-2009-5-257448 (2011).

[27] P. Eugster, P. Felber, R. Guerraoui, A. Kermarrec, The many faces of publish/subscribe, ACM Computing Surveys (CSUR) 35 (2) (2003) 114–131.

[28] D. Clark, R. Braden, A. Falk, V. Pingali, Fara: Reorganizing the addressing architecture, in: ACM SIGCOMM Computer Communication Review, Vol. 33, ACM, 2003, pp. 313–321.

[29] K. Katsaros, G. Xylomenos, G. Polyzos, A hybrid overlay multicast and caching scheme for information-centric networking, in: INFOCOM IEEE Conference on Computer Communications Workshops, 2010, IEEE, 2010, pp. 1–6.

[30] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, et al., Oceanstore: An architecture for global-scale persistent storage, ACM Sigplan Notices 35 (11) (2000) 190–201.

[31] R. Moskowitz, P. Jokela, P. Nikander, T. Henderson, Host identity protocol, RFC 5201 (2008).

[32] J. Rajahalme, M. Särelä, K. Visala, J. Riihijärvi, On name-based inter-domain routing, Computer Networks 55 (4) (2011) 975–986.

[33] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, S. Surana, Internet indirection infrastructure, in: ACM SIGCOMM Computer Communication Review - Proceedings of the 2002 SIGCOMM, Vol. 32, ACM, 2002, pp. 73–86.

[34] M. D'Ambrosio, P. Fasano, M. Marchisio, V. Vercellone, M. Ullio, Providing data dissemination services in the future internet, in: Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE, IEEE, 2008, pp. 1–6.

[35] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, P. Nikander, Lipsin: line speed publish/subscribe internetworking, in: SIGCOMM '09 Proceedings of the ACM, Vol. 39, ACM, 2009, pp. 195–206.

[36] M. Särelä, T. Rinta-aho, S. Tarkoma, Rtfm: Publish/subscribe internetworking architecture, in: ICT-MobileSummit 2008 Conference Proceedings, 2008, pp. 1–8.

[37] D. Trossen, Conceptual architecture: Principles, patterns and sub-components descriptions, Tech. rep., FP7-INFSO-ICT-257217 (2011).

[38] A. Detti, N. Blefari-Melazzi, Network layer solutions for a content-centric internet, Trustworthy Internet (2011) 359–369.

[39] D. Mazieres, M. Kaminsky, M. Kaashoek, E. Witchel, Separating key management from file system security, in: Proceed-

ings of the 17th ACM symposium on Operating systems principles, Vol. 33, ACM, 1999, pp. 124–139.
[40] M. Walfisha, H. Balakrishnana, S. Shenkerb, Untangling the web from dns, Networked System Design and Implementation (NSDI).
[41] W. Chai, I. Psaras, M. Charalambides, G. Pavlou, W. Chai, I. Psaras, Interim specification of mechanisms, protocols and algorithms for the, Tech. rep., FP7-2010-ICT-248784-STREP (2000).
[42] D. Trossen, G. Parisis, Architecture definition, components, descriptions and requirements, Tech. rep., FP7-INFSO-ICT-257217 (2011).
[43] A. Ericsson, Netinf content delivery and operations, Tech. rep., P7-ICT-2009-5-257448 (2012).
[44] P. Cao, S. Irani, Cost-aware www proxy caching algorithms, in: Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, Vol. 193, 1997.
[45] M. Charikar, S. Guha, Improved combinatorial algorithms for the facility location and k-median problems, in: 40th Annual Symposium on Foundations of Computer Science, 1999, IEEE, 1999, pp. 378–388.
[46] S. Khan, I. Ahmad, Comparison and analysis of ten static heuristics-based internet data replication techniques, Journal of Parallel and Distributed Computing 68 (2) (2008) 113–136.
[47] L. Qiu, V. Padmanabhan, G. Voelker, On the placement of web server replicas, in: INFOCOM 2001. Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 3, IEEE, 2001, pp. 1587–1596.
[48] S. Zaman, D. Grosu, A distributed algorithm for the replica placement problem, IEEE Transactions on Parallel and Distributed System 22 (9) (2011) 1455–1468.
[49] V. Sourlas, P. Flegkas, G. Paschos, D. Katsaros, L. Tassiulas, Storage planning and replica assignment in content-centric publish/subscribe networks, Computer Networks 55 (18) (2011) 4021–4032.
[50] S. Borst, V. Gupta, A. Walid, Distributed caching algorithms for content distribution networks, in: INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–9.
[51] J. Kangasharju, J. Roberts, K. Ross, Object replication strategies in content distribution networks, Computer Communications 25 (4) (2002) 376–383.
[52] B. Li, M. Golin, G. Italiano, X. Deng, K. Sohraby, On the optimal placement of web proxies in the internet, in: INFOCOM'99 Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 3, IEEE, 1999, pp. 1282–1290.
[53] S. Jamin, C. Jin, A. Kurc, D. Raz, Y. Shavitt, Constrained mirror placement on the internet, in: INFOCOM 2001. Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 1, IEEE, 2001, pp. 31–40.
[54] M. Korupolu, M. Dahlin, Coordinated placement and replacement for large-scale distributed caches, IEEE Transactions on Knowledge and Data Engineering 14 (6) (2002) 1317–1329.
[55] M. D'Ambrosio, C. Dannewitz, H. Karl, V. Vercellone, Mdht: a hierarchical name resolution service for information-centric networks, in: Proceedings of the ACM SIGCOMM workshop on Information-centric networking, ACM, 2011, pp. 7–12.
[56] J. Choi, J. Han, E. Cho, T. Kwon, Y. Choi, A survey on content-oriented networking for efficient content delivery, Communications Magazine, IEEE 49 (3) (2011) 121–127.
[57] L. Fan, P. Cao, J. Almeida, A. Broder, Summary cache: a scalable wide-area web cache sharing protocol, IEEE/ACM Transactions on Networking (TON) 8 (3) (2000) 281–293.
[58] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, H. M. Levy, On the scale and performance of cooperative web proxy caching, in: Proceedings of the 7th ACM symposium on Operating systems principles, SOSP '99, ACM, New York, NY, USA, 1999, pp. 16–31. doi:10.1145/319151.319153.
[59] J. Kjällman, N. Fotiou, Progress report of component implementations, Tech. rep., FP7-INFSO-ICT-257217 (2012).
[60] D. Rossi, G. Rossini, Caching performance of content centric networks under multi-path routing (and more), Tech. rep., Relatório técnico, Telecom ParisTech (2011).
[61] D. Rossi, G. Rossini, On sizing ccn content stores by exploiting topological information, in: Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2012, IEEE, 2012, pp. 280–285.
[62] W. Chai, D. He, I. Psaras, G. Pavlou, Cache "less for more" in information-centric networks, 11th International IFIP TC 6 Networking Conference, Prague, Czech Republic, May 21-25, 2012, Proceedings 7289 (2012) 27–40.
[63] G. Pavlou, N. Wang, W. Chai, I. Psaras, Internet-scale content mediation in information-centric networks, Annals of Telecommunications (2012) 1–11.
[64] S. Arianfar, P. Nikander, J. Ott, On content-centric router design and implications, in: Proceedings of the Re-Architecting the Internet Workshop, ACM, 2010, p. 5.
[65] I. Psaras, W. Chai, G. Pavlou, Probabilistic in-network caching for information-centric networks, in: Proceedings of the 2nd edition of the ICN workshop on Information-centric networking, ACM, 2012, pp. 55–60.
[66] G. Kamel, N. Wang, A. Beben, J. Sliwinski, J. Batalla, P. Wisniewski, W. Burakowski, W. Chai, I. Psaras, J. Araújo, et al., Final specification of mechanisms, protocols and algorithms for enhanced, Tech. rep., FP7-2010-ICT-248784-STREP (2007).
[67] N. Laoutaris, H. Che, I. Stavrakakis, The lcd interconnection of lru caches and its analysis, Performance Evaluation 63 (7) (2006) 609–634.
[68] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, S. Pack, Wave: Popularity-based and collaborative in-network caching for content-oriented networks, in: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2012, IEEE, 2012, pp. 316–321.
[69] A. Anand, V. Sekar, A. Akella, Smartre: an architecture for coordinated network-wide redundancy elimination, in: Proceedings of the ACM SIGCOMM 2009 Conference on Data communication, Vol. 39, ACM, 2009, pp. 87–98.
[70] H. Che, Y. Tung, Z. Wang, Hierarchical web caching systems: Modeling, design and experimental results, IEEE Journal

on Selected Areas in Communications 20 (7) (2002) 1305–1314.

[71] M. Rabinovich, I. Rabinovich, R. Rajaraman, A. Aggarwal, A dynamic object replication and migration protocol for an internet hosting service, in: Proceedings of the 19th IEEE International Conference on Distributed Computing Systems, 1999, IEEE, 1999, pp. 101–113.

[72] T. Wong, G. Ganger, J. Wilkes, C.-M. U. P. P. S. O. C. SCIENCE., My cache or yours? making storage more exclusive, Tech. rep., School of Computer Science, Carnegie Mellon University (2000).