Towards On-path Caching Alternatives in Information-Centric Networks

Andriana Ioannou
School of Computer Science and Statistics
Trinity College
Dublin 2, Ireland
Email: ioannoa@scss.tcd.ie

Stefan Weber
School of Computer Science and Statistics
Trinity College
Dublin 2, Ireland
Email: Stefan.Weber@scss.tcd.ie

***Abstract*—Information-Centric Networking (ICN), an alternative to the current Internet architecture, focuses on the distribution and retrieval of content instead of the transfer of information between specific endpoints. Approaches to ICN employ caches in the network to eliminate the transfer of information over lengthy communication paths from a source to consumers.**

The contribution of this paper lies in the placement of copies in on-path in-network caching. Our goal is to investigate the suitability of a probabilistic algorithm, *Prob-PD*, based on two variables, the content's popularity rates and the distance ratio of each node from the source, with regard to caching performance, i.e. cache hit rates, cache replacement rates and content delivery times. Towards this goal, we present an initial comparison of simulation results of the proposed caching mechanism and published alternatives showing significant gains of the algorithm.

***Index Terms*—Network Distributed Architectures; Future Internet; Information-Centric Networks; Caching technologies; Content replication; On-path caching**

I. Introduction

Information-Centric Networking (ICN) is an alternative to the end-to-end communication paradigm of the current Internet architecture, based on the publish-subscribe model [12], that focuses on content distribution and retrieval. In a publish-subscribe model, content sources make their content available by publishing it to a content notification service, i.e. a name resolution service or a name-based routing service, while clients request content from a content notification service by subscribing to it. In contrast to existing content distribution technologies, Content Delivery Networks (CDNs) for content replication and Peer-to-Peer (P2P) networks for file sharing, ICN networks are free of the application restrictions and the commercial boundaries [35] that the previous ones apply.

ICN architectures identify content resources, such as web pages, files or parts of a content resource, chunks or packets, using a content identifier; object, chunk or packet naming granularity. Ideally, content identifiers should involve no information that would bind the content to a specific location [5], [36]. If this constraint is met, the content can be freely replicated, therefore, provided by more than one source. Approaches to caching can be categorized into *off-path caching* and *on-path caching* with regard to the location of caches [2], [37]. Both caching approaches can be applied either separately or as a combination. The benefits of on-path caching against the off-path caching are still under investigation [11], [14].

Off-path caching, also referred as content replication or content storing, aims to replicate content within a network regardless of the forwarding path. Off-path caching is usually centralized and involves a great amount of information collected as well as advertised into a content notification service. The ICN off-path caching problem is equivalent to the CDN content replication and the web cache placement problems [32], [37]. As such, existing algorithms can be used [16], [18].

On-path caching on the other hand, is integrated to the architecture itself, i.e. the caching decision is limited to the content propagated along the delivery path(s) and to the nodes lying on the delivery path(s), caching is accomplished at the network layer, on a chunk level or a packet level, thus, being independent of the application used, caching mechanisms must follow the on-line speed requirements of the delivery process where the overhead of monitoring, collection of statistical information or advertisement of the cached content into a content notification service may not be acceptable or feasible. In addition to this, on-path caching does not follow any structural model; the topology is considered arbitrary.

In this paper, we focus on the efficiency of caching mechanisms for the on-path caching problem identified in the area of ICN. Towards this goal we propose a probabilistic algorithm, *Prob-PD*, based on two variables, the content's popularity ratio *(*P), and the distance ratio of each node from the source *(*D), which we compare against the alternatives via simulations.

The remainder of this paper is structured as follows. In section II, we describe the related work in the area of on-path caching and conclude to the most efficient algorithms. In Section III, we identify the gaps of the related work and describe the *Prob-PD* algorithm. In Section IV, we provide the details of the simulation model and the evaluation results of the approach against the most efficient alternatives concluded in Section II, as well as indications for future work. We close the paper with Section V, dedicated to the conclusions.

II. Related Work

Due to its integration with the architecture and the requirements deriving from it, on-path caching has triggered the interest of the research community, resulting in a number of

TABLE I: Description of the evaluation metrics of the related work.

Evaluation Metric	Description
Server hit rates	No. of content requests satisfied by the server
Cache hit rates	No. of content requests satisfied by a cache
Eviction rates	No. of cache replacements occured in a cache
Absorption times	Total time for which a content stays cached in the system
Hop count rates	No. of hops that a content request travels before being satisfied
Download times	Total time to retrieve a content

proposals. In this section, a summary of the proposed on-path caching approaches and their evaluation results is provided, with regard to a set of performance metrics such as the server hit rates and the absorption times. A detailed description of the performance metrics can be found in Table I.

A noticeable percentage of on-path caching algorithms has been proposed according to which a node *n* on the delivery path, decides to cache a content based on a probability *p* [4], [21]. We call these algorithms *Probabilistic*. Probabilistic algorithms can be categorized depending on the way that the caching probability is calculated; based on a pre-determined value [4], [10], therefore, called *FIX(p)* or based on a mathematical formula [27]. In a *FIX(p)* approach, *p* may be decided randomly or determined based on the number of nodes of the delivery path [8], [10]. The last category of *FIX(p)* algorithms is called *UniCache* due to the fact that the caching probability is uniformly distributed among the nodes. Considering an example of four intermediary nodes, the caching probability at each node would be *p=1/4*. The CE^2 algorithm, proposed by Jacobson et al. [15], is a *FIX(p)* algorithm with *p=1*. *FIX(p)* algorithms are lightweight algorithms, involving no co-operation between the nodes or information collection. However, they are unable to exploit any knowledge about the content or the network topology, resulting into low performance gains and high redundancy; based on the information provided in Table II, *FIX(p)* algorithms have been proven to be less efficient compared to other on-path caching approaches [8], [27], in terms of cache hit rates, hop count rates and download times.

The *FIX(p)* algorithms have been also tested against the *LCD* algorithm [30], an algorithm proposed by Laoutaris et al. [21] for the multilevel web caching problem, as an alternative to the de facto CE^2 approach. The *LCD* algorithm caches a copy of the requested content one hop closer to the client each time a content request arrives. According to the results provided by Rossi et al. [30], the *LCD* algorithm results into lower gains compared to the *FIX(p)* and CE^2 approaches with regard to the cache hit rates. Recent studies have used the *LCD* algorithm in combination with an exponentially inceasing content caching approach [10]. The examination, though, of such an algorithm is out of the scope of this paper as every on-path caching algorithm is able to be combined with it.

In contrast to the *FIX(p)* algorithms, Psarras et al. [27] have suggested a probabilistic caching algorithm that they name *ProbCache*, composed by two factors, the *TimesIn* factor and the *CacheWeight* factor. The *TimesIn* factor is calculated based on the capacity of the remaining nodes along the delivery path. According to the authors, the *TimesIn* factor favors contents that travel from further away while the *CacheWeight* factor acts as a counter-balance to this unfairness. Psarras et al. [27] have evaluated their algorithm against the *FIX(p)*, CE^2 and *LCD* alternatives, indicating significant gains in terms of server hit rates, hop count rates and eviction rates. However, the suitability of the algorithm is still an open question as some information about the parameters used on the experiments is omitted, among which, the number of contents; the number of contents is a critical design issue that defines the variance of the contents and hence the necessity of them to be cached.

Sourlas et al. [33] have proposed an on-path caching algorithm called *LeafNode*. According to this approach, the content is cached at the last node of the delivery path. Sourlas et al. [33] have evaluate the *LeafNode* algorithm against the CE^2 one, resulting into higher hop count rates and lower absorption times than its alternative. Similar to other works, however, the evaluation results conluded are rather questionable since no information about the number of contents or the content size is provided. *LeafNode* caching has been also proposed by Katsaros et al. [19] as the supporting caching method for their MultiCache overlay architecture. However, the evaluation of their work is against the BitTorrent application, providing no information about the performance of the caching algorithm.

As caching is performed at the network layer of the architecture, graph-based metrics may be used for deciding the node to which caching should take place. Chai et al. [8] have proposed an on-path caching algorithm based on the metric of *Betweeness Centrality (BC)*; the BC metric represents the number of times that a node *n* lies at the sets of shortest paths, between all the pairs of nodes in the graph, besides *n*. According to the algorithm, caching is performed only at the nodes holding the highest *BC* value. Chai et al. [8] have proven that the *BC* approach provides lower server hit rates and lower hop count rates than the CE^2 and *UniCache* approaches.

Rossi et al. [31] have examined the suitability of a number of graph-based algorithms, *Degree Centrality (DC)*, *Betweenness Centrality (BC)*, *Closeness Centrality (CC)*, *Graph Centrality (GC)*, *Eccentricity Centrality (EC)* and *Stress Centrality(SC)*, for determining the size of the caches, e.g. proportional to the centrality value of each node rather than determing the nodes for caching the content. Based on their evaluation results, Rossi et al. [31] have conclude that *DC*, which indicates the number of edges of each node *n*, is the most effective graph-related metric compared to the alternatives, in terms of server hit rates and hop count rates.

TABLE II: Taxonomy of the proposed on-path caching algorithms.

Proposed Technique	Comparison Technique	Caching model	Evaluation Metrics	Comparison Results	Topology type
BC [8]	UniCache, CE^2	centralized	server hit rates, hop count rates	BC > CE^2> UniCache	CAIDA (6804 nodes)
CE^2[15]	-	autonomous	-	-	-
DC, BC, CC, GC, EC, SC [31]	DC, BC, CC, GC, EC, SC	centralized	cache hit rates, hop count rates	DC > SC, BC, CC, GC, EC	Rocketfuel (up to 68 nodes)
FIX(p) [4]	CE^2	autonomous	cache hit rates, latency	CE^2> FIX(0.5) > FIX(0.3) > FIX(0.25) > FIX(0.125)	8-nodes string
LCD [30]	CE^2, FIX(p)	autonomous	cache hit rates, hop count rates, load fairness, cache diversity	FIX(0.90) > FIX(0.75) > CE^2> LCD	Rocketfuel (up to 68 nodes)
LeafNode [33]	CE^2	autonomous	hop count rates, absorption time	CE^2> LeafNode	up to 4-level binary tree
ProbCache [27]	CE^2, LCD, FIX(p)	dependent	server hit rates, hop count rates, eviction rates	ProbCache > LCD, FIX(0.70) & FIX(0.30) > CE^2	6-level binary tree

A general disadvantage deriving from the nature of graph-based caching algorithms is their limited efficiency due to their centralized nature; centralized information is required for their operation, collected by e.g. a topology manager. Therefore, graph-based algorithms are improper for inter-domain scale; even under an optimistic scenario where the Administrative Systems (ASes) would exchange traffic information to conclude to a unique centrality value, the scalability issues of a global topology manager would be an open question. The *DC* algorithm constitutes an exception to this rule. In addition to this, graph-based algorithms operate on a specific set of nodes, leaving the rest of the nodes on the delivery path, unexploited.

Table II summarizes the proposed on-path caching approaches and the ones against which they have been compared to, as well as, the metrics and the topologies under which they have been evaluated and the evaluation results. The approaches may be further categorized regarding the information that they use for the caching decision, i.e. *autonomous caching*; using local information, *centralized caching*; using centralized information and *dependent caching*; using information regarding other nodes in a non-centralized manner. Whether a cached content is propagated into a content notification service; *caching-aware system* or not; *opportunistic-caching system* and the level of operation, *object-level, chunk-level* or *packet-level* to which each caching mechanism refers to is irrelevant to our analysis. In this table, the symbol *"-"* indicates that no information has been provided regarding this category while the symbol "$a > b$" indicates that approach *a* results in better performance than approach *b*.

Concluding this section and based on the summary presented in Table II, three approaches, the *FIX(0.90)*, *DC* and *ProbCache*, seem to outperform the rest of the alternatives. One of the main contributions of this work is the evaluation of these algorithms against each other. Based on this comparison and the previous ones performed by the research community, we expect to conclude to the nature of the caching system that would be more beneficial for an ICN architecture. We expect that *dependent* caching algorithms, such as the *ProbCache* approach, may perform better than the alternatives.

III. Probabilistic-PD Algorithm

Based, on the information summarized at the related work section, section II, one can easily observe the absence of the content popularity as a criterion to the on-path caching decision. We argue, that content popularity is an important factor, able to affect the performance of a caching algorithm and the network as a whole [7], [30]. Therefore, it should be taken into account. Content popularity has been applied on a number of cache replacement policies and replication algorithms proposed for the object replication problem, defined in the areas of web proxies and CDNs, e.g. [16], [18], [26].

Measurement studies in web caching have highlighted the problem of *cache pollution* due to *one-timer* objects [26]. One-timer objects is a term used to identify objects that are requested only once while cache pollution is a term used to identify the case where one-timer objects are cached, decreasing the performance of the caching mechanism; cache pollution prevents the caching of more popular objects, resulting in an increase with regard to the cache miss rates and the network traffic rates. According to the same studies, one-timer objects correspond to approximately 45-75% of the total requests. As on-path caching is expected to serve a much higher number of contents than object replication mechanisms, under more severe restrictions, such as the cache capacity restrictions [4], [23], the prevention of the cache pollution problem becomes an even more important prerequisity.

Towards this direction and inspired by the *Local Greedy* algorithm, proposed by Kangasharju et al. [18] for content replication on CDNs and the lack of consideration of the content popularity into the on-path caching decision, we propose a probabilistic algorithm, called *Prob-PD*, consists of two factors, the content's popularity ratio *P*, observed on a node and the distance ratio between the same node and the source serving the content, *D*. The idea behind popularity-based caching is that more popular contents will satisfy more requests. Therefore, caching popular contents should be preferred. At this point one should make a decision regarding the way that content popularity is caclulated. Based on this criterion, content popularity may be divided into *static-content*

popularity and *dynamic-content popularity*.

Static-content popularity approaches require the definition of a threshold h. Contents with a number of requests higher than h are considered to be popular while contents with a number of requests lower than h are considered to be unpopular [9], [17], [25], [29]. Unpopular contents are excluded from the caching decision. Due to the volatile nature of ICN architectures, we expect the definition of h to be challenging; static-content popularity approaches usually result into out of date calculations and unutilized cache capacity [17]. Therefore, applying a dynamic-content popularity approach on an ICN architecture, instead of a static one, should be preferred.

Dynamic-content popularity is defined as the number of requests for a content during a time interval Δ_t [22], [34]. Consequently, the popularity of a content is concluded by comparing its request rates against each other's. A common technique to provide an up to date content popularity pattern is to sort the contents in a decreasing order [6], [13], [18]. A disadvantage deriving from this technique is the constant comparison of the request rates. Towards this direction, we propose a dynamic-content popularity approach that reduces the number of comparisons to a minimum [3].

We have already stated the reasons why we chose popularity to be one of the factors that construct our caching algorithm. We now attempt to explain the reasons why we chose distance to be the combined factor. Similarly to other caching techniques, on-path caching is requested to answer the question of where to cache a content. We slightly change this question into whether a content should be cached on a node or not, based on a network-related criterion. Latency reduction is probably one of the most preferable goals with regard to network performance therefore one of the most known network metrics. The number of hops has been defined as a good estimation of the latency metric, being used in routing protocols and CDN replication algorithms, e.g. [18], [20], [24], [28].

In order to explain the caching algorithm further, we first define some notations. Let i denote the node performing the caching decision and j denote the content on which the caching decision is applied. Let $r_{i,j}$ denote the number of requests recorded on node i for content j, with $\sum_{j=1}^{J \leq Ch} r_{i,j}$ being the total number of requests and $d(i,i')$ the distance between nodes i and i', using the number of hops as a metric. We then define the $Prob-PD_{i,j}$ algorithm as follows:

$$Prob-PD_{i,j} = \underbrace{\frac{\frac{r_{i,j}}{\Delta_t}}{\sum_{\forall j \epsilon Ch} \frac{r_{i,j}}{\Delta_t}}}_{P} \times \underbrace{\frac{d_{n,src}}{d_{dst,src}}}_{D} \quad (1)$$

where, *P* is the dynamic-popularity calculation of content *j*, constructed by the number of requests for content *j* during the time interval Δ_t, divided by the total number of requests during the same time interval, on node *i*. In order to avoid introducing any additional overhead to the operation of a node, we define Δ_t to be the time between the arrival of the first request for content *j* and the satisfaction of it. This way, a content is limited to one comparison only against the rest of the contents, minimizing the complexity that dynamic-popularity calculations apply. The *D* factor is constructed based on the distance between node *i* and the source *src* serving the request, normalized by the distance between *src* and the node requesting the content *dst*. The *D* factor represents the benefit of caching the content on the current node against the cost of retrieving the content from the original source. Our goal is to examine how beneficial may be the combination of these two factors regarding the ICN on-path caching problem.

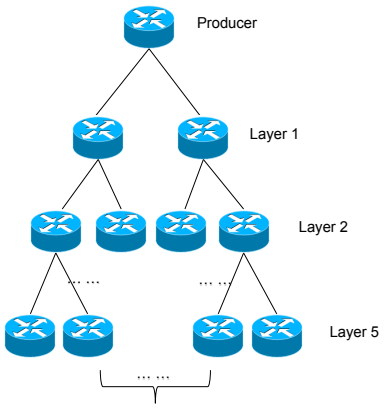

Fig. 1: 5-level binary-tree evaluation topology.

IV. Evaluation

In order to provide some initial indications of the performance of the proposed algorithm, we include the results of some early experiments based on the *ndnSIM* simulator [1], an ns-3 module that implements the Named Data Networking (NDN) communication model [15]. A summary of the evaluation model is presented in Table III. The experimental topology, Fig.1, is a 5-level binary-tree, with the root being the source of *1000* contents. All nodes, except the root, are able to request a content and cache it in their content store *(CS)*. The capacity of a *CS* is equal to 10 contents. Contents in a *CS* are replaced using a LRU policy [4], [15], [30]. Content requests are generated on each node, in parallel, with a constant frequency, following a Zipf distribution of $\alpha \epsilon \{1.0, 1.5\}$. Last, a shortest path routing protocol is assumed while the capacity and the delay of the connection links being equal to 100Mbps and 10ms, respectively. The results are concluded after the completion of approximately $3x10^7$ content requests in total.

Using the simulation model described, we compare the performance of the CE^2, *DC*, $FIX(0.90)$, $ProbCache$ and $Prob-PD$ algorithms with regard to the overall cache hit rates, overall cache replacement rates and average content delivery times. Due to space restrictions, we modify the names of two of the approaches into a shorter abbreviation, i.e. *PC* and *PD*, for the ProbCache and Prob-PD algorithms, respectively. The results are presented on Fig.1 for *a=1.0* and on Fig.2 for *a=1.5*. On each figure, the x-axis corresponds to the nodes of the binary-tree topology while the y-axis corresponds to the evaluation metric recorded, e.g. the cache hit rates. One general observation that can be derived from

TABLE III: Parameters of the system model used for evaluation.

Parameter	Symbol	Value	Definition
No. of nodes	N	63	Total No. of nodes
No. of producers	P	1	Total No. of producer nodes
Capacity of links	BW	100Mbps	Available bandwidth
Link delay	D	10ms	Link delay
No. of contents	C	1000	Total No. of contents in chunks
Cache Size	CS	10	Cache capacity of a node in chunks
Consumers Size	u_i	1	No. of users on node i
Zipf Exponent	α	$\alpha\epsilon\{1.0, 1.5\}$	Exponent of the Zipf popularity distribution
Control window	W	∞	No. of requests able to sent with no reply

the aforementioned figures is a certain pattern of anomalies on the behavior of the algorithms, e.g. the lower content delivery times, with regard to the simulation topology; one can easily distinguish that the points where these anomalies appear are the same points where the level of the binary-tree updates. Further investigation on this behavior is out of progress.

According to Fig.1, the PD algorithm seems to outperform the rest of the approaches in terms of cache hit rates and cache replacement rates. More precisely, the DC algorithm is scaling in similar rates after the 30^{th} node, with regard to the cache hit rates while the PC algorithm is scaling slightly worse, with regard to the cache replacement rates. As expected, the CE^2 and $P(0.90)$ algorithms result in the lowest cache hit rates and the highest cache replacement rates but they still hold the advantage of the lowest delivery times. However, the same, roughly, delivery times are reached by the PD algorithm. The aforementioned results verify our claim that dependent caching algorithms may perform better than the autonomous ones.

Comparing Fig.1 and Fig.2, one can conclude that the general trend of the algorithms is preserved for the majority of them, as the a increases from 1.0 to 1.5. The most noticeable difference is the behavior of the PC algorithm, resulting in higher cache hit rates and higher content delivery times. The distinctive behavior of the DC algorithm after the 30^{th} node is related to the nature of the algorithm; a content is cached, only at the nodes having the highest number of neighbors along the delivery path. Therefore, nodes 1^{st}-30^{th}, having a DC value equal to three, would be preffered over nodes 31^{st}-62^{nd}, having a DC value equal to one. As such, nodes 31^{st}-62^{nd} would have a lower chance of caching a content originated from the source. However, this also means that the nodes would have more free space so as to cache contents being found in the 3^{nd} level of the binary-tree hierarchy, resulting in higher cache hit rates and lower replacement rates.

A final point that we owe to highlight is the dependance of the performance of the *Prob-PD* algorithm, *PD* in this section, to the nature of the workload, such as the number of contents. This drawback derives from the way that the popularity factor *P* is calculated. Considering an example where the number of requests for content *j* during the time interval Δ_t is 10 and the total amount of requests during the same time interval is equal to 1000, then $P = 0.01$, which would probably result in a non-caching decision of the content. We do recognize that this validation may be important and a different approach to the way popularity is calculated should be considered.

To this end, an alternative option is about to be tested, where *P* is equal to the number of requests for content *j* during the time interval Δ_t, divided by the maximum number of requests observed for any content, during the same time interval. The advantage of this approach lies to the fact that contents are now competing with each other and not with the whole number of requests. Following the same example as before and assuming that the highest number of requests for any content observed is equal to 50, then $P = 0.2$. That way a better approximation of the content's popularity may be provided. However, the idea may not be beneficial in terms of calculation resources. An alternative to this would be the usage of a predefined time interval. We attempt to fulfill this as a future work.

V. Conclusion

In this paper, we have described the existing on-path caching algorithms for the ICN architectural model and categorized them against their properties. We have further proposed a probabilistic caching algorithm, *Prob-PD*, to enhance performance. In order to have some initial results, we evaluated our approach against the CE^2, *DC*, $FIX(0.90)$ and *ProbCache* alternatives. The results indicate that *Prob-PD* may provide significant gains if certain conditions are met. However, since our model is an early evaluation, further research is necessary to conclude to the suitability of the algorithms and their performance.

References

[1] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim: Ndn simulator for ns-3," *Technical Report NDN-0005, Named-Data Networking Project*, October 2012.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[3] F. M. Al-Turjman, A. E. Al-Fagih, and H. S. Hassanein, "A value-based cache replacement approach for information-centric networks," in *Proceedings of the 13th IEEE International Workshop on Wireless Local Networks (WLN '13), Sydney, Australia, October 2013*, pp. 874–881.

[4] S. Arianfar, P. Nikander, and J. Ott, "On content-centric router design and implications," in *Proceedings of the Re-Architecting the Internet Workshop of the ACM CoNEXT Conference*, no. 5, 2010.

[5] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish, "A layered naming architecture for the internet," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications (SIGCOMM '04), Portland, OR, USA, August 2004*, pp. 343–352.

[6] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proceedings of the 29th IEEE Conference on Computer Communication (INFOCOM '10), San Diego, CA, USA, March 2010*, pp. 1–9.

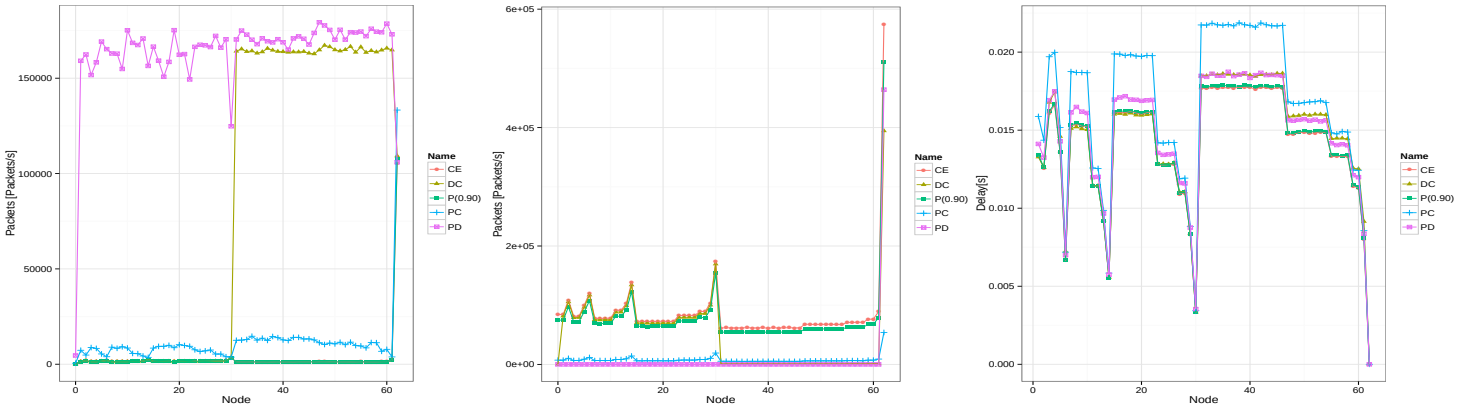

Fig. 2: Metrics for $\alpha = 1.0$, $(a.)$ Cache hit rates, $(b.)$ Cache replacements rates, $(c.)$ Content delivery times

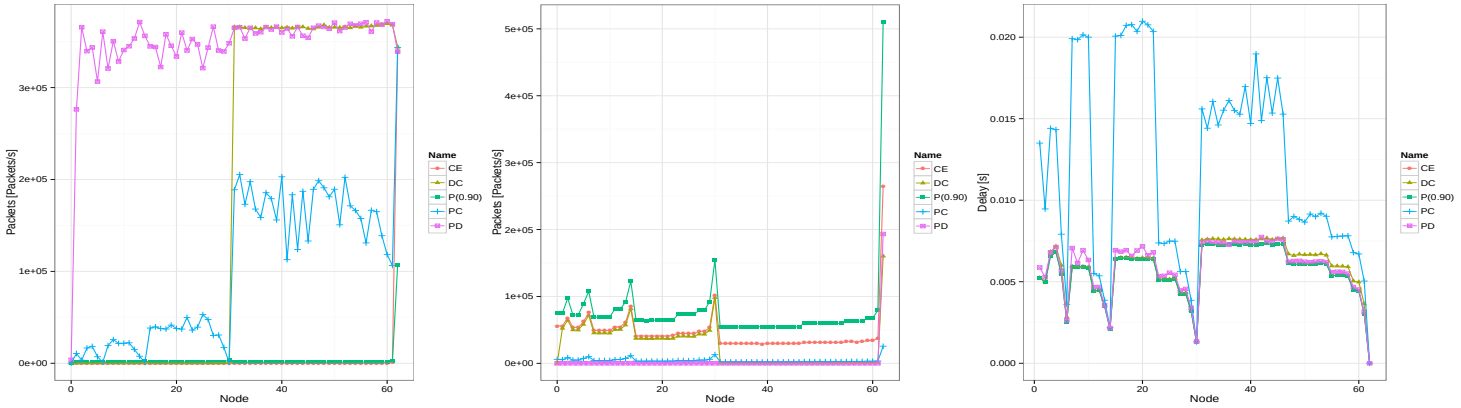

Fig. 3: Metrics for $\alpha = 1.5$, $(a.)$ Cache hit rates, $(b.)$ Cache replacements rates, $(c.)$ Content delivery times

[7] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in content-centric networking," in *Proceedings of the 23rd International Teletraffic Congress (ITC'11), San Francisco, CA, USA, September*, 2011, pp. 111–118.

[8] W. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," in *Proceedings of the 11th International IFIP TC 6 Conference on Networking, Brooklyn, NY, USA, May 2012*, pp. 27–40.

[9] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, 2002.

[10] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *Proceedings of the 1st Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN '12), Orlando, FL, USA, March 2012*, pp. 316–321.

[11] M. Draxler and H. Karl, "Efficiency of on-path and off-path caching strategies in information centric networks," in *Proceedings of the IEEE International Conference on Green Computing and Communications (GreenCom '12), Besancon, France, Novemver 2012*, pp. 581–587.

[12] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003.

[13] N. Fujita, Y. Ishikawa, A. Iwata, and R. Izmailov, "Coarse-grain replica management strategies for dynamic replication of web contents," *Elsevier Journal on Computer Networks*, vol. 45, no. 1, pp. 19–34, 2004.

[14] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: seeing the forest for the trees," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X '11), Cambridge, MA, USA, November 2011*, p. 1.

[15] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '09), Rome, Italy, December 2009*, pp. 1–12.

[16] S. Jamin, C. Jin, A. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," in *Proceedings of the 20th IEEE Conference on Computer Communication (INFOCOM '01), Anchorage, AK, USA, April 2001*, pp. 31–40.

[17] T. Janaszka, D. Bursztynowski, and M. Dzida, "On popularity-based load balancing in content networks," in *Proceedings of the 24th International Teletraffic Congress (ITC'12), Krakow, Poland, September 2012*, pp. 1–8.

[18] J. Kangasharju, J. Roberts, and K. Ross, "Object replication strategies in content distribution networks," *Elsevier Journal on Computer Communications*, vol. 25, no. 4, pp. 376–383, 2002.

[19] K. Katsaros, G. Xylomenos, and G. Polyzos, "Multicache: An overlay architecture for information-centric networking," *Elsevier Journal on Computer Networks*, vol. 55, no. 4, pp. 936–947, 2011.

[20] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 568–582, 2000.

[21] N. Laoutaris, H. Che, and I. Stavrakakis, "The lcd interconnection of lru caches and its analysis," *Elsevier Journal on Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006.

[22] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, "Distributed selfish replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 12, pp. 1401–1413, 2006.

[23] U. Lee, I. Rimac, and V. Hilt, "Greening the internet with content-centric networking," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (e-Energy '10), Passau, Germany, April 2010*, pp. 179–182.

[24] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the internet," in *Proceedings of the 8th IEEE Conference on Computer Communication (INFOCOM '99), New York, NY, USA, March 1999*, pp. 1282–1290.

[25] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong, "Popularity-driven coordinated caching in named data networking," in *Proceedings of the 8th ACM/IEEE Symposium on Architectures for*

Networking and Communications Systems (ANCS '12), Austin, TX, USA, October 2012, pp. 15–26.

[26] A. Mahanti, D. Eager, and C. Williamson, "Temporal locality and its impact on web proxy cache performance," *Elsevier Journal on Performance Evaluation*, vol. 42, no. 2, pp. 187–203, 2000.

[27] I. Psaras, W. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the 2nd Workshop on Information-Centric Networking, Orlando, FL, USA, March 2012*, pp. 55–60.

[28] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of web server replicas," in *Proceedings of the 20th IEEE Conference on Computer Communication (INFOCOM '01), Anchorage, AK, USA, April 2001*, pp. 1587–1596.

[29] M. Rabinovich, I. Rabinovich, R. Rajaraman, and 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 (ICDCS '99), Austin, TX, USA, May 1999*, pp. 101–113.

[30] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech Ecole, Tech. Rep., November 2011.

[31] G. Rossini and D. Rossi, "On sizing ccn content stores by exploiting topological information," in *Proceedings of the 1st Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN '12), Orlando, FL, USA, March 2012*, pp. 280–285.

[32] V. Sourlas, P. Flegkas, G. Paschos, D. Katsaros, and L. Tassiulas, "Storage planning and replica assignment in content-centric publish/subscribe networks," *Elsevier Journal on Computer Networks*, vol. 55, no. 18, pp. 4021–4032, 2011.

[33] V. Sourlas, G. Paschos, P. Flegkas, and L. Tassiulas, "Caching in content-based publish/subscribe systems," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '09), Honolulu, HI, USA, November 2009*, pp. 1–6.

[34] V. Sourlas, P. Flegkas, L. Gkatzikis, and L. Tassiulas, "Autonomic cache management in information-centric networks," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS '12), Maui, HI, USA, April 2012*, pp. 121–129.

[35] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Journal on Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.

[36] M. Walfisha, H. Balakrishnana, and S. Shenkerb, "Untangling the web from dns," in *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI '04), San Francisco, CA, USA, March 2004*.

[37] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys and Tutorials*, no. 1–26, 2013.