A Multi-Path based In-network Content Caching Scheme

Deepak Bhatia
University of Surrey
Guildford, UK
d.bhatia@surrey.ac.uk

Ning Wang
University of Surrey
Guildford, UK
n.wang@surrey.ac.uk

Michael Howarth
University of Surrey
Guildford, UK
m.howarth@surrey.ac.uk

Abstract – In this paper we propose an efficient scheme for supporting multi-path based in-network content caching by leveraging Loop-Free Alternate (LFA) in IP routing. The purpose is to enable clustered distribution of cached content chunks at en-route routers between the content source and the requester. In order to achieve caching localization, we also introduce policies that limit the maximum allowable distance between caching routers and the final destination, which is known as caching radius. The benefit of the scheme is to distribute contents locally around interested receivers in a balanced manner. We evaluated the proposed algorithm through simulations based on the GEANT network topology. The performance exhibited by the multi-path based caching scheme outperforms conventional approaches based on single path routing.

I. INTRODUCTION

The consumption of dramatically increasing volumes of content objects in the Internet has stimulated research efforts into the re-design of the underlying network architecture which is expected to be content-oriented. In this case, the primary effect of content consumption has put content at the center of future design of the Internet. The rapid growth of available content in the Internet has resulted in the birth of several content delivery mechanisms, such as CDNs, P2P and more recently ICNs (information centric networks). Compared to the traditional application-layer paradigms, ICNs are deemed to be more revolutionary that put content level knowledge and intelligence at the network layer. A typical example is that en-route routers have the capability to store/cache content objects as they are being delivered. Most well-known ICN architectures include CCN/NDN [7], DONA [8] and PSIRP/PURSUIT in the literature.

In-network content caching has been a hot research topic in the context of ICN research. The idea is to cache popular content items within the network between the content server and receivers. The WAVE scheme [3] focused on efficiently populating network caches chunk-by-chunk of content to improve content service performance. WAVE adopts a window-based algorithm that is executed locally at each router along the path between the content server and the client. At each step, based on the popularity the number of chunks (window) to be cached is increased. Its execution starts close on directing content flows symmetrically. The authors of [4] proposed to use age of content measured by the distance from the source of content, as a metric to achieve caching closer to destinations. It uses age of content to decide whether content can be cached and replaced. Content age is longer if its distance from the server is higher. The longer age also indicates higher popularity of content. Thus more popular content can thus be stored closer to the destination router or the edge of the network.

In this paper, we introduce a new in-network caching scheme that allows content chunks to be cached in a clustered manner along multiple paths towards interested receivers. The motivation is to enable more routers nearby to have the opportunity of being involved in caching content locally. We achieve this by ensuring that the caching operation is initiated by the neighbouring nodes of the receiver and then it further expanded to more remote routers. In this case, upstream routers can only cache content chunks if they are aware that downstream routers that are closer to the content client have no cache space available. This is achieved based on the signalling communication between routers along each content delivery path. We also introduce the policy that allows the content client to pose a caching radius around itself. Such a radius indicates the maximum allowable distance between the farthest router that can participate in content caching for this requester. A small radius means that fewer routers can participate in content caching for the requester. The second novelty of our approach is to employ multiple paths in plain IP based networks to deliver content chunks to content clients. As far as multi-path routing is concerned in CCN, the authors in [1] lay the responsibility of determining multiple paths on the node that generates the initial requests. Requests are then forwarded simultaneously on these multiple paths and content chunks are subsequently directed on these paths, by routers that have requests entries made in their Pending Interest Table (PIT). These multiple paths have no common paths between them. In contrast to [1], in our design multiple paths are identified between requesting node by application of Loop-Free Alternate (LFA) technique which has already been standardized in IP routing. In addition, we do not propose to direct content chunks simultaneously on multiple paths as is the case in [1]. The proposed multi-path content directing and caching algorithm creates clusters of content around routers that generated the initial request. Clusters are formed by restricting caching and applying LFA only within a pre-configured distance from destination router. Our focus is on understanding the effectiveness of caching chunks of content in clusters around each requesting node.

The technical contribution of this paper is summarised as follows. First of all, the proposed algorithm creates efficient *content clusters* around destination routers (clients) for local content access. We propose a novel in-network content caching algorithm that leverages on Loop-free alternate (LFA) protocol to identify multiple paths on which contents are delivered and cached.
The efficiency of the proposed algorithm is evaluated based on the European GEANT network topology in a realistic simulation environment. The results have indicated the benefit of using multiple paths instead of single paths for delivering and caching content chunks and avoid the negative effects of sending identical contents on multiple paths as is the case in [1].

Rest of the paper is organized as follows. Section II describes the system design and the algorithm which is explained with an example. Section III description of the simulation environment and the results are presented. Section IV describes the conclusions that we arrive at based on results of the experiments.

II SYSTEM DESIGN

The main objective of caching content along multiple paths between source and requesting client is to enable content availability in clusters closer to interested users. The reasoning for multiple paths against single path is to allow more content routers near the client to get involved in content caching. In general, identifying multiple paths must not entail any additional communication overhead between routers.

A. Identifying multiple paths based on LFA

First of all, it should be noted that the responsibility of identifying multiple paths lies with routers en-route from destination to server on which request is directed. These en-route caches are called intermediate routers. Computation of alternate paths to other routers is performed by each intermediate router independently in an offline manner. These computations utilize the Loop Free Alternate (LFA) path concept [2] which is originally for IP fast reroute purpose. Specifically, LFAs are computed by routers to ensure uninterrupted flow of network traffic in case any link failure between two nodes. Router computes dedicated LFA next-hop neighbours towards different destinations. The standard LFA is based on inequality condition below [2] in the context of Fig. 1.

$$dist(o, u) + dist(u, d) > dist(o, d)$$

More specifically, consider the head node u in the figure which is responsible for computing alternate LFA paths towards the destination d. The default shortest path from u to d is via link l and the end-to-end distance between the node pair is dist(u, d). Node o (the neighbouring node of u) enables a feasible alternative LFA path if the distance from o to u plus the distance from u to d is larger than the distance from o to d. In this case, if the head node u "deflects" any packet towards d onto neighbour o which is not on the default shortest path, node o is still able to natively forward the packet to d along its own shortest path but without returning the packet back to u.

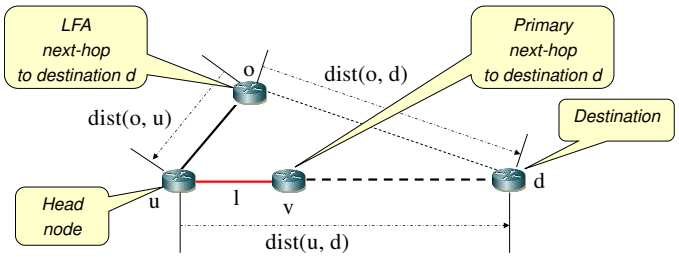

Fig 1.Loop Free Alternate Path

Forwarding content chunks along alternate paths is referred to as deflecting in this piece of work. Application of LFA to multi-paths caching is based on utilizing these LFAs to identify multiple paths between intermediate nodes and the content clients as final destination on the request path. Content chunks are forwarded along these paths so routers that are sufficiently close to the client can cache chunks flowing towards destination. The design permits LFAs to be used starting from routers closest to the destination routers, this is to ensure caches closer to destination form content clusters. It is noted that once a content chunk has been deflected, it cannot be deflected again in order to avoid forwarding loops. Another rule enforced is to prevent routers that are further than pre-determined limit of hops from the destination router (i.e. out-of-radius routers) from directing along LFAs.

B. Proposed Algorithm

During the operation, content requests are sent by each client router towards the targeted content source. Each request contains a hop counter, i.e. indicating the hop counts that request has travelled. Initially, client node that generates the request i.e. destination router sets the hop counter to 0 and thereafter it increases by 1 after traversing each hop towards the content source. This updated information is stored at each router that has received the content request. Such information is used by the intermediate routers to determine whether they are located within or outside the radius from the client.
The proposed algorithm is executed at each intermediate router with local caching capability, when a piece of incoming content chunk is received. The algorithm has two main components, decisions on caching content chunks and forwarding chunks along alternate paths to enable caching chunks on en-route intermediate routers. Only the routers along the default shortest paths are permitted to deflect content chunks along alternate LFA paths towards destination routers. Content requests are generated for individual content chunks R(C) and sent towards the source/s. As individual requests are received at routers, an entry is created in the request table (RT). Entries in the request table are used to identify content chunks that need to be deflected on alternate LFA paths. The algorithm has two procedures Store-and-Forward and Send-on-LFA_{DR} (see the pseudo code on the right). The Store-and-Forward procedure of the algorithm is the main function,

where the algorithm execution begins. It is initialized by checking the content chunk C received on the interface IC_R has been requested by comparing entry for C in request table RT, which is done in Step 1. Next, the algorithm compares the distance of the current node to the destination router (DR) in Step 2. If the current node is outside the radius, it can be forwarded on shortest path to destination, as shown in Step 12. This is to ensure only routers that are close to destination router can apply LFA and cache. In order to ensure routers close to the destination cache first or only cache it if the downstream router has sent a cache full message, these comparisons are performed in Step 3. The decision to cache is based on the unavailability of LFAs, Step 4 and 5. Step 4 calls the procedure **Send-on-LFA$_{DR}$** that checks for availability in Step 1 of the procedure. Every available LFA path is checked for any received cache full message in Step 3. If no cache full message has been received, the content C does not need to be cached and it is forwarded on the LFA and the algorithm execution is stopped at Step 7 of procedure **Send-on-LFA$_{DR}$**. If there are no LFA paths available or a cache full message is received on each LFA, a true value is returned in Step 10. The **Store-and-Forward** procedure next checks the returned value in Step 5, if it is true and not previously cached based on valued of pCache. The Step 16 of the procedure is executed if the router is not on the request path. It checks to see if the next hop is the destination or a cache full message has been received. The content is cached if the conditions in Step 16 are fulfilled; else they are forwarded on shortest path to destination. If the content is cached at any router, it is marked as cached and forwarded downstream towards destination.
The algorithm finally checks the condition of the cache in Step 23, if it is full to capacity it sends a cache full message upstream on the interface the chunk was received. The cache full message also contains the distance of the router from the destination client. This makes sure a router which is not on the default shortest path to determine whether it is within or outside the specified caching radius from the client.

Multiple Path Content Chunk Caching Algorithm
C: index of chunk of Content C
CR: Curent Router
DR: Destination router
LFA$_{DR}$: Alternate Path to D from Current Node.
RT: Request Table
R(C): Request for Content chunk C.
d(DR): Distance from current router to destination router.
P: Path
ID$_R$: Interface i on shortest path to DR from CR.
CF$_P$: Cache Full message on path P.
CF: Cache Full message indicating distance to DR from CR.
Size$_{CR}$: Size of Cache at current router.
Capacity$_{CR}$: Capacity of Cache at current router.
IC$_R$: Receiving Interface for content C.
pCache: Boolean variable indicating content chunk has already been previously cached at an upstream router.
pDeflected: Boolean variable indicating content has been deflected i.e. sent on LFA.

```
Procedure Store-and-Forward
1: if R(C) ∈ RT then
2:     if d(DR) <=Radius then
3:             if CF_P ≠ null or d(DR)= = 1 then
4:                     cache = procedure Send-on-LFADR
5:                     if cache = = true and pCache = = false then
6:                                 Store C in cache
7:                                 pCache = true
8:                     endif
9:                         forward on shortest path P
10:                endif
11:            else
12:                forward on shortest path P
13:    endif
14:endif
15: else
16:    if d(DR)= = 1 or CF_P ≠ null and pCache = = false then
17:                Store C in Cache
18:                pCache = true
19:    endif
20:           forward on P
21: if Size_CR = = Capacity_CR then
22:    Send CF message on IC_R
23: endif

Procedure Send-on-LFA_DR
1:  cache = true
2: if pDeflected  = = false then
3:         for every P∈ LFADR
4:                 if CF_P  = = null then
5:                   cache = false
6:                   pDeflected=true
7:                   forward on P
8:                   stop algorithm
9:                 endif
10:        endfor
11: endif
12: return cache
```

Now we further explain the mechanism of the cache full message used in the protocol. The Cache full message is a unique approach to communication among intermediate routers along default and LFA paths. They indicate that downstream router cannot store more contents in their cache. A cache reaches a cache full state, if all the contents in its cache have been requested within a specified time period. The time period is also preconfigure by the ISP. The replacement policy applied is LRU. In comparison to traditional application of LRU, we only remove if the least recently used item has not been used within the last few time units. The design assumes routers store distance information once they have received a cache full message with the distance to a destination.

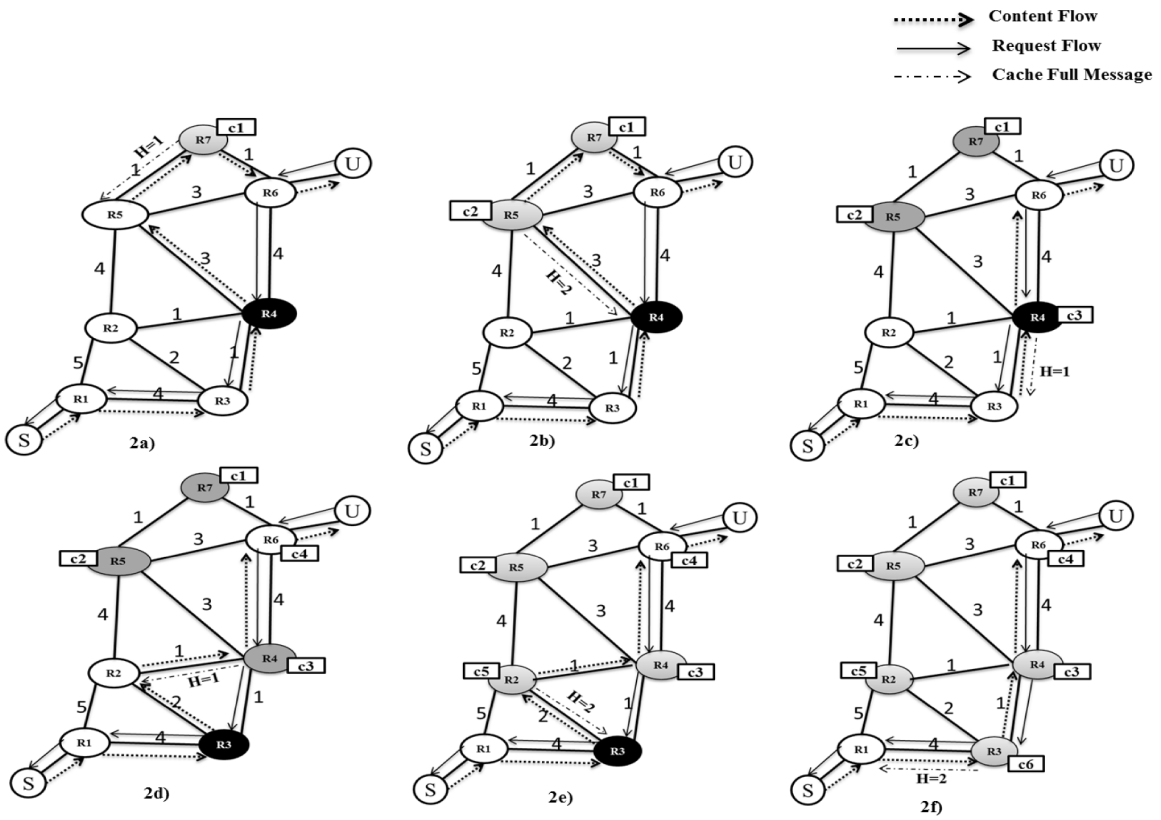

Fig 2. Explaining multi-path chunk caching scheme

C. An example

In this section, we will give a demonstration of the algorithm described above by using a simple example. The example demonstrates the delivery of content chunks from server S to the content client U. In this case the destination router in Fig.2 is router R6 where U is attached. Initial requests for content are received at R6. For simplicity we make an assumption that all intermediate routers have one chunk capacity for caching. In each step, routers marked in black are responsible for deflecting content chunks onto alternate LFA paths. Cache capacity of routers marked grey is full and cannot cache any more contents. Thus they send out cache full messages to routers upstream. Although it is not shown in the example, the content requests are sent from R6 towards S on the shortest path i.e. R6-R4-R3-R1 according to the IGP link weight shown in the topology (next to each link). R6 sets the hop counter to 0, as it reaches R4 it is increased to 1 and this value is added to the requests and forwarded towards R3. These hop counter values are thus stored at intermediate routers and forwarded along with the requests. As, can be seen, in Fig. 2a router R4 is marked black and thus instead of caching the chunks, it first deflects them along the LFA to R6 i.e. R4-R5-R7-R6. R5 then forwards the chunks along its own shortest IGP path to R6. As the scheme design is to allow closest routers to start caching, R4 should have cached, but as there is LFA to R6, it directs it on that path. Thus, R7 is the first router to cache first chunk c1. Due to the assumption that cache capacity is one, cache space at R7 is full and it sends a cache full message back to its previous hope R5. Next, when R4 forwards the 2nd chunk c2 to R5 along the LFA path, as the chunk is not marked as cached, R5 caches it as it knows that downstream R7 can no longer cache it. Subsequently, R5 sends a cache full message to previous hop R4. The next chunk c3 is then cached at R4, before being forwarded to R6 on the direct link. After caching R4 has no more space left to cache any more content and hence it sends a cache full message to its previous hop R3 as indicated in Fig. 2c. In Fig. 2d, R3 deflects future content chunks flowing through it i.e chunk c4 on LFA toward destination R6 via its neighbour R2 and further to R4 which is on the shortest path to R6. But as R4 cache is full, it does not cache c4 and thus the chunk arrives at the last-hop node R6 which has to cache it, as can be seen in Fig. 2e. Once router R3 has received cache full message from R2, with the hop count included, the next chunk c6 is cached at R3. As R3 is full it sends cache full message upstream towards R1 as can be seen from Fig. 2f. Thus, to emphasize a cache full message is only sent on interface on which content chunk that was cached has been received. In this example it is assumed that the request is completed by the server. The similar actions would apply if the requests would be completed by any en-route cache.

III. SIMULATION RESULTS

Table 1 represents the parameters that are applied in our simulation experiments. We simulate over the GEANT topology. The total number of available content items is 10^8, with each chunk 10 KB. The cache size at each router is 1GB.

Request generation uses Mandelbrot-Zipf model with α,q={2.0,50}. Previous research has also focused on applying this model for request generation [3][4]. These values are used as we focus of video content like Youtube, VoD etc. which exhibits request pattern with α=2.0[7]. The experiments were conducted based on the following number of content servers {1, 5, 10, 15} randomly placed within the network. Content requests are randomly generated from all nodes in the topology. As such, different numbers of alternative LFA paths may exist for specific locations of content server-client pairs. The cache replacement policy used is Least Recently Used(LRU). The statistics are collected for contents that form 90% tile of request, so that the behavior of algorithm for very popular contents is the main focus. The simulations results are averaged over 5 independent runs each. The results are presented with a 5% error value. Table 2 shows the different test cases we have implemented and tested. In order to test the performance of the application of LFA, we test caching on all en-route routers, applying LFA within the radius = 1. We differentiate the cache and LFA radius to enable highlighting this case.

Table 1.Experiment Parameters

Parameter	Values
Network topology	GEANT (23 nodes and 74 links)
Total number of content items	10^8
Size of each content	100 chunks
Size of each chunk	10 KB
Cache Size	1 GB
α,q	{2.0,50}
Number of content servers	{1, 5, 10, 15}
Caching radius considered	{1, 2, 3}

Our algorithm aims at creating clustered content chunk caches nearby by allowing caches within a given radius to store contents. Naturally, it is necessary to compare against the approaches that allow caching everywhere along the route. Another important aspect is towards understanding the benefit of applying Loop-free Alternate (LFA) to create content clusters. Additionally, we study the effect of "breadth" of content clusters and thus tests results for different radius parameters.

Table 2.Test Cases

Server number	Cache Radius	LFA Radius
1	1,2,3	1,2,3
5	1	1
10	1	1
15	1	1
1	No	No
1	No	1

A. Performance Metrics

Cache Hit Ratio: The cache hit ratio is defined as ratio of total number of content chunk requests that can be served from the en-route caches rather than from the original server over the total number of content requests received. It measures the utility of caching policy applied aggregated over the entire topology.

Path Length Ratio: It is calculated as the ratio of the hop count from the cache hit router back to the receiver and the end-to-end hop count from the original server to that receiver. Path Length Ratio can be directly used for evaluating the localisation efficiency of content caching scheme.

Cache Efficiency: This metric is defined as the percentage of the content chunks that have been requested since they have been cached locally in a router against the total number of chunks in the cache. With this metric we want to show the fairness of the applying caching policy across the entire topology.

B. Performance

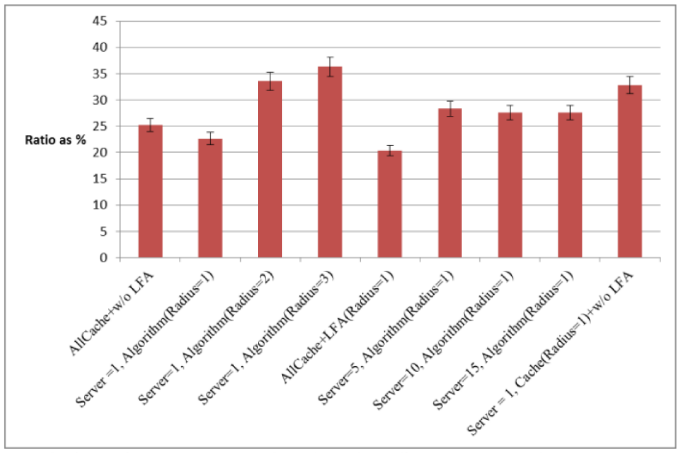

Fig 3. Performance comparison based on distance to complete requests

LFA deflection is applied only by routers that are located within the pre-determined radius value from the destination. This in combination with caching within the radius ensures creating content clusters around each receiver. Also, as a content chunk has once been deflected, further downstream routers along the path does not deflect it again. The results in Fig.4 indicate behaviour of the algorithm in terms of finding content in cache is the best for the radius=1 scenario, with the value at 22.5% of distance to the source, at which we can find content chunk in caches to complete the request. This value increases to 32.5% and 45% at radius values 2 and 3 respectively. The results are intuitive, as with higher radius values, the chunks may travel over longer distance towards the destination. When the number of servers is increased, it may be expected to find content closer, but as larger number of routers cache content chunks en-route, the caches store significantly more content items that are not accessed after they are cached, they are replaced more frequently and thus the probability of finding the content in caches is lower. Thus increases the distance at which content can be found in cache. The benefit of LFA can be seen from comparing the result against all caches en-route store contents with or without LFA. Applying LFA reduces the distance to 20%, while without LFA the distance value increases to 25%. The results can be verified from Fig. 5 that shows higher cache hit at 65% compared to 53% for all en-route caches storing content but

without applying LFA. Fig. 3 presents the comparison of the performance of the algorithm against the different cases stated above for the GEANT topology. Caching in clusters around the destination the algorithm achieves 20% higher performance compared to caching at all routers en-route when LFA is used but experiences a significant drop in cache hit ratio (being the worst performer by 20%) when LFA is not applied. Cache hit ratio performance achieved at radius=2 and 3 is poorer compared to radius=1 by 20% and 33%, and this can also be verified from Fig. 4. Thus, percentage of copies available inside a network reduces at higher radius due to evictions. Future requests have to be completed by servers or routers located further from destination router. The number of content copies being stored is higher as at radius=3, number hops used are higher compared to not applying LFA. The results for placing multiple servers inside the network when our algorithm is applied are not as promising as in a single server case at radius 1. Larger number of servers increases the percentage of contents that are available at shorter distances. This increases the percentage of contents that would be stored in caches, thus increasing the probability of increase in the number of replacements.

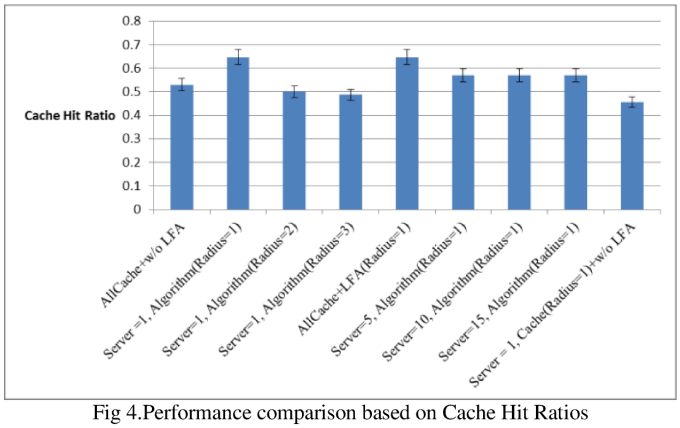

Fig 4.Performance comparison based on Cache Hit Ratios

Caching efficiency plotted in Fig. 5 proves that the algorithm works better at lower radius values. As explained in the section on metrics, efficiency is computed based utility of the cached content chunk locally to the router. The higher the value the better is the performance of the algorithm. In case of a single server the performance is on average 50% higher for radius=1 compared to that of radius=2, 3. The results show the viability of applying LFAs for use in in-network caching at lower radius values. Instead of focusing a single path or multiple parallel paths(that lead to identical content being stored on entire length of those paths, resulting in higher proportion of cache removals and thus achieve significantly lower cache hit ratios[1]), algorithm provides a controlled approach to pushing content on multiple paths.

IV. CONCLUSION

In this paper we proposed an algorithm to design a multiple path content caching strategy which is a novel approach of applying Loop free Alternate (LFA) to identify multiple paths. The multi-path strategy is leveraged to create cluster of content around routers that cache content en-route. Clusters are created by enforcing a pre-determined radius that can be configured by an ISP. En-route router/caches within the radius can store content chunks in caches and deflect those on LFAs. The scheme was evaluated for single and multi-server scenarios through simulations on top of the GEANT network topology. Performance achieved demonstrated the advantage of using multiple paths, when using them in a limited radius i.e. distance from the destination. At larger radius values, multiple caches suffer from significant overlap of contents, thus limiting the advantage due to higher percentage of contents that are not accessed after being store in local caches. With the application of intelligent routing, it is possible to achieve load balancing and traffic optimizations among caches and complete future requests at shorter distances.

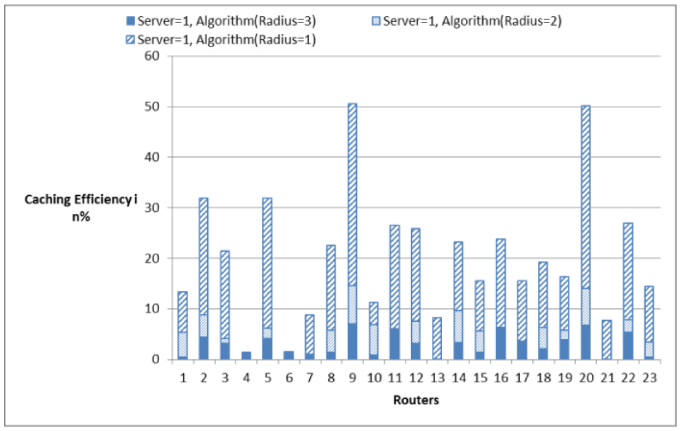

Fig 5.Caching Efficiency Comparison

Acknowledgments

This paper is partially funded by EU FP7 IRSES EVANS Project (PIRSES-GA-2010-269323) and FP7 COMET Project (FP7-2010-ICT-248784).

REFERENCES

[1] D. Rossi, G. Rossini, “Caching performance of content-centric network under multi-path routing (and more)”, Technical report, Telecom ParisTech, 2011.

[2] A. Atlas, A. Zinnin, “Basic Specification for IP Fast Reroute: Loop-Free Alternates”, RFC 5286, Sep 2008.

[3]Cho, K., Lee, M., Park, K., Kwon, T. T., Choi, Y., and Pack, S.,"WAVE: Popularity-based and Collaborative In-network Caching for Content-Oriented Networks," In Proceedings of INFOCOM’12 NOMEN Workshop, 2012.

[4]Zhongxing Ming, Mingwei Xu, Dan Wang. “Age-based Cooperative Caching in Information-Centric Networks", In Proceedings of INFOCOM’12 NOMEN Workshop, 2012.

[5]I. Poese, B.Frank, B. Ager, G. Smaragdakis, and A. Feldmann., “Improving content delivery using provider-aided distance information”, In Proceedings of ACM IMC, 2010.

[6]H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, “P4P: Provider portal for applications”,In Proceedings of ACM SIGCOMM, 2008.

[7]T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica,“A Data-Oriented (and Beyond) Network Architecture”, In Proceedings of ACM SIGCOMM, 2007.

[8]V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, R. Braynard, “Networking Named Context“, In Proceedings of ACM CoNext, 2009.