COntent Mediator architecture for content-aware nETworks

European Seventh Framework Project FP7-2010-ICT-248784-STREP

Deliverable D5.2
Scalability of COMET System

The COMET Consortium

Telefónica Investigación y Desarrollo, TID, Spain
University College London, UCL, United Kingdom
University of Surrey, UniS, United Kingdom
PrimeTel PLC, PRIMETEL, Cyprus
Warsaw University of Technology, WUT, Poland
Intracom SA Telecom Solutions, INTRACOM TELECOM, Greece

For more information on this document or the COMET project, please contact:

Andrzej Bęben
Warsaw University of Technology, abeben@tele.pw.edu.pl

Document Control

Title: Scalability of COMET System

Type: Public

Editor(s): Andrzej B□ben

E-mail: abeben@tele.pw.edu.pl

Author(s): Andrzej B□ben, Wojciech Burakowski, Aneta Fabja□ska, Jordi Mongay Batalla (WUT), George Kamel, Ning Wang (UNIS), Ioannis Psaras, Wei Koong Chai (UCL), David Flórez (TID)

Doc ID: d5.2_v1.0.docx

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
v0.1	24/09/12	Andrzej B□ben	ToC of D5.2
v0.2	20/11/12	Ioannis Psaras, Wei Koong Chai	Contribution about content caching
v0.3	23/11/12	George Kamel	Contribution about evaluation of coupled approach
v04	3/12/12	Andrzej Beben, Jordi Mongay Batalla	Contribution about large-scale Internet model and analysis of decision algorithms for ICN networks
v0.5	17/12/12	Wojciech Burakowski, Andrzej B□ben, Aneta Fabja□ska	Contribution about scalability evaluation
V0.6	4/01/13	Andrzej B□ben	Contribution about performance evaluation of RAE
V0.7	14/01/13	George Kamel	Updated contribution about evaluation of coupled approach
V0.8	30/01/13	Ioannis Psaras, Wei Koong Chai	Updated contribution about evaluation of content caching
V0.9	08/02/13	David Florez , Yiannis Psaras, Jordi Mongay Batalla Andrzej B□ben	Deliverable review
V0.95	13/02/13	Andrzej B□ben	Pre-final version
V0.96	15/02/13	David Florez , Yiannis Psaras, Andrzej B□ben	Final screening
V1.0	18/02/13	David Florez, Andrzej B□ben	Final version

Table of Contents

1 Executive Summary 5
2 Introduction 8
3 Large-scale model for scalability and performance evaluation 10
3.1 Introduction 10
3.2 The model of video content consumption 10
3.2.1 Network topology 10
3.2.2 Content server location 10
3.2.3 Content characteristic and distribution 11
3.3 Summary 11
4 Scalability and performance of the COMET decoupled approach 12
4.1 Introduction 12
4.2 Content resolution process 12
4.2.1 The simulation model 15
4.2.2 Evaluation of the basic COMET system 20
4.2.3 Evaluation of the extended COMET system 25
4.3 Content delivery process 30
4.3.1 Performance of packet forwarding 30
4.3.2 Analysis the length of forwarding key list 31
4.4 Conclusions 33
5 Scalability and performance of the COMET coupled approach 35
5.1 Introduction 35
5.2 Content Resolution Modes 35
5.3 Performance Evaluation 36
5.3.1 Simulation Setup 36
5.3.2 Empirical Results 36
5.4 Conclusions 39
6 In-network caching 40
6.1 Introduction 40
6.2 Probabilistic In-network Caching 40
*6.2.1 Extending **ProbCache** for Heterogeneous Cache Sizes 41*
6.2.2 Summary of (Preliminary) Evaluations 43
6.2.3 Analysis I 43
6.2.4 Maximum Values of Basic Functions 44
6.2.5 Density Distribution of Basic Functions 46
*6.2.6 Enhancing the content multiplexing fairness of **ProbCache** 47*

6.2.7 Maximum Values of Enhanced Function *48*
6.2.8 Density Distribution of Enhanced Functions *49*
6.2.9 Summary of Theoretical Analysis and Findings *50*
6.2.10 Performance Evaluation *51*
6.3 Centrality-based In-network Caching 57
6.3.1 Model Description *57*
6.3.2 Performance Metrics and Evaluation Methodology *58*
6.3.3 A Preliminary Test *59*
6.3.4 Performance in k-ary Tree Topologies *59*
6.3.5 Performance in Scale-free Topologies *61*
6.3.6 Performance in Real AS-level Topologies *63*
6.3.7 Caching Operation Overhead *64*
6.4 Conclusions 65
7 Performance of multi-constrain and multipath routing 67
7.1 Mullti-constraint and multipath routing 67
7.2 Simulation experiments 68
7.3 Results and conclusions 69
8 Multi-criteria decision algorithm for ICN networks 71
8.1 Validation of decision algorithms in dynamic environment 72
8.2 Performance evaluation 74
8.2.1 Request arrival process *74*
8.2.2 Routing protocols *74*
8.2.3 Decision strategies *75*
8.2.4 Simulation results *75*
8.2.5 Notes about the trustworthiness and reliability of the results *78*
8.3 Conclusions 78
9 Summary and conclusions 80
10 References 82
11 Abbreviations 86
12 Annex A: Paper submitted to IEEE TPDS 88
13 Annex B: Paper submitted to Computer Networks 89
14 Annex C: Paper submitted to Springer JTS 90

1 Executive Summary

This deliverable focuses on scalability and performance evaluation of the COMET system. COMET designed two complementary approaches: the coupled and the decoupled. Both of them are currently intensively investigated by leading research groups working on Information Centric Networks (ICN). The *decoupled* approach was designed to deal with the majority of Internet content, while the *coupled* approach was designed to support widely accessed popular content, which can benefit from in-network caching. In the decoupled approach, content resolution takes place first, followed by content access using the server identified through the resolution process [10], [11]. In the coupled approach, content resolution and access are combined in a single phase, with content resolution following a gossip-like communication model, routing content consumption requests in a specific manner within the mediation plane in order to locate the targeted content source [12].

In chapter 2, we introduce the scalability problem and we specify the main questions:

1. Assuming our best knowledge about the characteristics of the content requests (i.e., what content, request rate), what performance can we expect from the COMET system?
2. What happens when the system size (i.e., request rate, number of contents, number of servers, number of domains) increases?
3. Where are the bottlenecks in the COMET system and what are the barriers in system extension, if any?

In chapter 3, we introduce a large-scale network model defined for performance and scalability evaluation of the COMET system. The model takes into account: (1) a large-scale model of Internet topology, (2) content server location and characteristics, and (3) content characteristics and distribution. The model of Internet topology is based on AS-level data sets provided by the CAIDA [14] from January 2011. We consider 3-level hierarchical Internet topology (Tier-1, 2 and 3) according to the business relationship between domains (provider-customer and peering relationships). The resulting topology has 36,878 domains, where approximately half of them are stub domains, and 206,969 inter-domain links. The content server location and characteristics are defined based on the analysis of the 50 largest video content providers and CDNs, which are, among others: Level(3), Global Crossing, LimeLight, Akamai, AT&T, Comcast and Google [20]. The features of the video content and distribution are based on data from [30]. We analysed the duration of the movies for 5,000 most popular titles. The results indicate that mean duration of the movie is around 4100 s. For each content, we attached a random value of streaming bandwidth comprised between 2.6 and 3.4 Mbps, approximating the range of bandwidth the videos in Netflix Canadian network are streamed [31].

Chapter 4 presents simulation studies on scalability and performance evaluation of the COMET decoupled approach. In these studies, we focus on evaluation of the content resolution and content delivery processes. The COMET system was modelled as an *open queuing network*, where particular COMET entities play the role of servers. The queuing network is fed by the arrivals of content requests. Each request initiates the sequence of tasks executed in different servers. The input data for this model were obtained from the experiments carried out on the COMET prototype deployed over the federated testbed. In our studies, we evaluated what is the impact of increased system size (request rate, number of content sources, domain size and number of domains) on the content resolution and content delivery processes. The obtained results confirmed that there is no critical element in the COMET architecture, which could lead to the system performance degradation. More specifically, we can state that:

- The execution delays of the content resolution process, even under relatively high load (0.8), are satisfactory for the customers because the Content Resolution Time (CRT) for 95 per cent of content requests is less that 2.5 s;
- Even when the system size increases (content request rate, number of content sources, domain size) the CRT delays are still acceptable;

- The overhead of the COMET header is in similar range as the IPv6 header even for the longest paths;
- The CAFE performance is similar to the performance of the IP router (based on measurements reported in D6.2 [9].

The results of our studies confirmed that there should be no barriers in deploying the COMET system in large-scale networks, like the Internet. Further improvements of the content resolution procedure are possible, e.g., by applying: (1) caching mechanisms in the Client CME for storing content/authoritative records, paths and servers' load and (2) smart querying algorithms for the retrieval of servers load and routing paths.

Chapter 5 presents simulation studies of the coupled approach, comparing the broadcast and random resolution schemes, and their effect on the number of ASes that are traversed to resolve a single content. Furthermore, we measure the path length of the final resolution path along which the content travels to reach the content client. We find that the final resolution path under both resolution modes is virtually the same on average, meaning that the higher number of AS traversals under Broadcast mode compared to Random mode leads to no justification for the use of Broadcast mode. This argument is further reinforced by the fact that the average hop count of both resolution modes *after* route optimisation is also virtually the same. It was also shown that the average hop count along the resolution and optimised delivery paths does not seem to vary with topology size. Therefore, looking at the deployment of the coupled approach at an Internet-scale (purely from a scalability point-of-view), the results suggest that using even a random resolution scheme, the delivery path length is kept to within five hops. Consequently, the content retrieval time can be kept to well under one second, in part because of the single round trip required by the coupled approach to resolve content.

Chapter 6 focuses on the performance evaluation of the two in-network caching strategies developed in COMET. Namely, the Probabilistic In-Network Caching algorithm, ProbCache, initially presented D4.2 [4] and in [43] and the Centrality-based approach to caching presented also in D4.2 [4] and in [43]. ProbCache is an algorithm that approximates the capability of paths to cache contents, based on path lengths, and multiplexes content flows accordingly. The ultimate goal of ProbCache is to utilise resources efficiently, reduce caching redundancy and in turn, network traffic redundancy. We have considered both homogeneous and heterogeneous cache sizes and have adjusted ProbCache to fit in both environments. We report savings of up to 20% in server hits; 7-8% in the number of hops to hit cached contents; and reduction by an order of magnitude in cache evictions, which directly translates to network traffic redundancy elimination by the same proportion. The second algorithm proposed here is a centrality-based in-network caching approach. Node centrality-based caching is based on the concept of betweenness centrality (Betw) so that content is only cached at the nodes having the highest probability of being on the path of any source-destination pair and therefore, having the highest probability of getting a cache hit along the content delivery path. Our design ensures the content always spreads towards content users and thus reduces the content access latency. We also proposed an approximation of it (EgoBetw) for scalable and distributed realization in dynamic network environments where the full topology cannot be known a priori. Based on our extensive simulations, we observed that Betw consistently achieves the best hop and server reduction ratios across topologies having different structural properties without being restricted by the operating conditions required by random caching strategy. Our results further suggest that EgoBetw approximates closely Betw in non-regular topologies (e.g., B-A graphs) and thus presents itself as a practical candidate for the deployment of this approach. We also show that the caching overhead of ubiquitous caching is more than 60% higher than our Betw. Thus, we conclude that indeed caching less can actually achieve more and that our proposed (Ego)Betw approach is a potential candidate for realizing this promise.

Chapter 7 presents studies focused on performance evaluation of multicriteria and multipath protocol performed by RAE. The objective is to evaluate the RAE performance and compare its effectiveness with the reference BGP-4 protocol. The presented results confirmed that RAE improves the coverage of COMET Class of Service (CoS) in a multi-domain network by

providing more content delivery paths that satisfy constraints than the BGP4 protocol. The gain from RAE comes from two new features: the multi-criteria path selection algorithm and the advertisement of multiple paths. The gain from the RAE depends on the constraint values. The maximum gain is visible for moderate constraints, which are in the middle of concatenated values related to the average path length. Moreover, we observe that the gain from the RAE increases in networks with large number of domains. This effect comes from the fact that the number of alternative paths significantly increases with number of domains. Consequently, the multicriteria and multi-path routing algorithm has higher probability of finding feasible paths than the BGP-4 protocol.

Chapter 8 proposes and evaluates the multi-criteria decision algorithm, which exploits two closely related processes. The first process, which operates offline, discovers multiple content delivery paths and gathers their respective transfer characteristics. The second process, which is invoked for each content request, combines available information about network and server condition for selecting the best content server and delivery path. The simulation results confirm that the two-level algorithm provides more information to the selection of server and path. This results in a higher percentage of satisfied content requests by improving utilization of network and server resources. When the number of content requests increases, the two-level algorithm makes feasible load balancing in both network and servers, avoiding or slowing down overload conditions. Load balancing is achieved also in situations of normal load, which is an interesting feature for network operators and content providers.

Chapter 9 summarizes and concludes the deliverable. From the point of view of the system scalability, it is important to note that in the decoupled approach, the control of the content resolution process is performed mainly by the client CME and the server CME, both of them located in the edge domains. So, if we have problems with handing content requests in a given domain, we can offload the COMET entities by adding new instances in this domain. This is very positive feature of the discussed system supporting its scalability. One can find some similarities with DiffServ architecture, which is regarded as a reference scalable solution. In DiffServ the main processes are performed by edge routers while the core routers perform only forwarding. When traffic grows, we simply add new edge routers without changing the core. The simulation studies of the coupled approach showed that the use of broadcast resolution does not offer any benefit over random-based resolution in terms of content delivery performance, making the latter more preferable. In addition, it was shown that the hop count does not vary considerably with topology size, making the coupled approach a purely feasible option from a scalability perspective, and hence resulting in worst-case content retrievals of well under one second.

2 Introduction

In this deliverable, we deal with scalability and performance assessment of the COMET system. Such evaluation is a very important issue since COMET was designed for deployment over the Internet, where a huge number of users have access to more than 10^{11} content objects. Therefore, in our studies we need to take into account: (1) a large-scale model of Internet topology, (2) content server location and characteristics, (3) rules for distribution of content replicas, and (4) content requests' distribution. The COMET system introduces new elements in the Internet, performing content mediation and delivery functions. These elements are deployed in consumer, server and transit domains and perform a number of processes, each of them requiring special signalling/messaging. The processes responsible for handling users' requests are executed on-line (in the order of seconds), while the processes related to provisioning such as content publishing, network routing awareness, content delivery path management are executed off-line. Summarising, the system under study is rather complex and introduces new elements which may affect scalability.

COMET designed two complementary approaches: coupled and decoupled. Both of them are currently intensively investigated by leading research groups working on Information Centric Networks (ICN). The *decoupled* approach was designed to deal with the majority of Internet content, while the *coupled* approach was designed to support widely accessed popular content which can benefit from in-network caching. In the decoupled approach, content resolution takes place first, followed by content access using the server identified through the resolution process [10], [11]. In the coupled approach, content resolution and access are combined in a single phase, with content resolution following a gossip-like communication model, routing content consumption requests in a specific manner within the mediation plane in order to locate the targeted content source [12].

Taking into account the essential differences between these approaches, we need different models for scalability and performance evaluation. Both approaches exhibit the required properties for achieving large-scale content consumption. In this report, we attempt to answer the following questions:

1. Assuming perfect knowledge of content request characteristics (i.e., which content, what request rate), what is the expected performance from the COMET system?
2. What happens when the system size (i.e., request rate, number of contents, number of servers, number of domains) increases?
3. Where are the bottlenecks in the COMET system and what are the barriers in system extension, if any?

The analysis of the obtained results will allow us to outline guidelines related to COMET deployment over the current Internet and new functionalities of the network, which will support the expected COMET performance. In order to evaluate the two COMET approaches, we use measurements from the experiments carried out in the COMET federated testbed and develop simulation tools as well as analytical models.

Apart from COMET scalability studies, we present results on performance evaluation that focus on the key functionalities of the COMET system. In particular, we evaluate in-network caching, multipath routing and multi-criteria decision algorithms. Although the motivation of this research comes from the problems encountered during the COMET system design, we believe that the obtained results are universal and can be applied to other ICN systems.

The organisation of this document is as follows. In chapter 3, we present a large-scale model for content consumption and inter-domain routing evaluation. This model constitutes a base for our scalability and performance studies. Chapter 4 focuses on scalability of the decoupled approach. We present a model for scalability evaluation, description of simulation experiments, obtained results, as well as guidelines for system improvements. In Chapter 5, we present scalability and performance evaluation of the coupled approach. In the next chapters, we focus on performance evaluation of key functionalities of content networks. Our approach to in-network caching is

evaluated in chapter 6. The multicriteria and multipath routing algorithms are evaluated in chapter 7. The multicriteria decision algorithms designed for COMET and other content networks are studied in chapter 8. Finally, chapter 9 summarises the deliverable and outlines conclusions.

In addition, Annex A, B and C include draft papers related to investigated topics that were recently submitted IEEE Transaction on Parallel and Distributed System, Elsevier Computer Networks and the Springer Journal on Telecommunication Systems.

3 Large-scale model for scalability and performance evaluation

3.1 Introduction

In this chapter, we present a large-scale network model defined for performance and scalability evaluation of the inter-domain routing and the COMET system. The proposed model for video content consumption covers: (1) a large-scale model of Internet topology, (2) content server location and characteristics, and (3) rules for distribution of content replicas. The model has been described in D3.2 [3] and published in [13] and [64].

3.2 The model of video content consumption

3.2.1 Network topology

The model of the Internet topology is based on AS-level data provided by the Cooperative Association for Internet Data Analysis (CAIDA) [14], [15] from January 2011. We consider 3-level hierarchical Internet topology (Tier-1, 2 and 3) according to the business relationship between domains (provider-customer and peering relationships). The resulting topology has 36,878 domains, where approximately half of them are stub domains, and 206,969 inter-domain links. We assume that consumer population in a given domain is proportional to the length of network prefixes advertised by this domain. Using data provided by UOregon [16] and RIPE [17], we created the histogram of the number of advertised prefixes, presented in Figure 1. Moreover, this data confirmed that Tier 3 domains advertise the majority of prefixes, while Tier-1 and Tier-2 domains advertised less than dozens of prefixes.

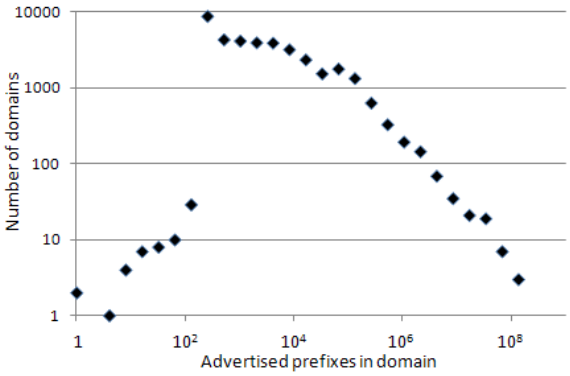


Figure 1: Histogram of advertised prefixes.

Finally, regarding link capacities, we observe that operators such as China Telecom and others currently connect to consumers with 1 Gbps links [19]. Therefore, we assume uniformly distributed probability density function U[0.5, 1.5] Gbps for links connecting Tier-3 domains. Furthermore, we assume the links connecting Tier-2 to its providers or peers to be ten times greater (i.e., U[5.0, 15.0] Gbps) and, finally, U[50.0, 150.0] Gbps links for connecting Tier-1 peering domains. Note that in our model these capacities are dedicated only to video traffic.

3.2.2 Content server location

For the analysis of the distribution of content servers within the network, we refer to the 50 largest video content providers and CDNs [20], [21], which are, among others: Level(3), Global Crossing, LimeLight, Akamai, AT&T, Comcast and Google The number of content servers in these domains corresponds to the information provided in public statements and white papers. Moreover, we identify that Akamai serves about 1,000 domains (as Youtube, QuickTime TV, etc.) with 84,000 multimedia servers [23]. For the remaining domains, we assign a random number of content

servers based on uniformly distributed probability density function U[50,150]. Finally, the total number of servers in the modelled network is approximately 200,000 servers.

The characteristics of the content server of interest are the maximum number of concurrent connections that may be served (limited by server disk I/O and network bandwidth [24],[25]) and the maximum number of contents stored. Current commercial servers may serve from 50 up to 1,000 concurrent connections [26],[27],[28]. For simplicity, we assume that all servers in the network have the same number of maximum concurrent connections but this number varies from one simulation to another. On the other hand, commercial servers differ much in storage capacity. A medium-size server may have 600 GBytes of content storage capacity [28], which approximately translates to 100 titles since one High Definition 2-hour movie may have a size of 5-8 GBytes [29]. Consequently, our model assumes that servers store in total around 20 million copies of content.

3.2.3 Content characteristic and distribution

The features of the video contents are based on data from [30]. We analysed the duration of the movies for the 5,000 most popular titles. The results indicate that mean duration of the movie is around 4100 s. For each content, we attached a random value of streaming bandwidth comprised between 2.6 and 3.4 Mbps, imitating the range of bandwidth the videos in Netflix Canadian network are streamed [31].

There are two possible techniques for assuring load balancing in content networks, which are striping and replication. Striping consists of partitioning the content between different servers, whereas replication consists of copying the content in several different servers. In most content replication approaches, the number of copies of given content depends on its popularity and it is widely accepted for video distribution in the Internet follows the Zipf's law [19],[24],[32]. In our model, the skew parameter of the Zipf's formula equals 0.2 as suggested in [19]. As explained above, the total number of copies stored in the servers is around 20 million. In order to have these copies (exactly 19,332,562 copies), the most popular content is copied 17,000 times and the rest follows the Zipf's formula. The distribution of copies in the servers is another key point in content networks [33], since a good distribution strategy may definitely increase the efficiency of the system. For simplicity, we assumed a random strategy subject to the condition that no more than one copy of given content may be in any server.

3.3 Summary

Table 1 presents the summary of presented above model parameters. Although the range of parameters to be modelled in video content Internet is broad, we believe our model reflects closely the reality.

Table 1: Parameters of the model for content distribution in the Internet.

Network Topology		
Number of domains	~36,000 domains	Sources: [16], [17]. About the half are stub domains
Number of links	~207,000 links	Sources: [16], [17]
Server characteristics		
Number of servers	~200,000 servers	Source: Akamai [23] and CDN
Capacity of servers	100 titles	Sources: [28] and [29]
Content characteristics		
Number of content files	5,000 titles	Source: Film Web Inc. [30]
Number of copies	20,000,000 copies	
Mean duration of content	4100 s	Source: Film Web Inc. [30]
Streaming bandwidth	U [2600,3400] kbps	Source: NetFlix [31]
Content popularity	Zipf's law (skew parameter= 0.2)	Source: [19]

4 Scalability and performance of the COMET decoupled approach

4.1 Introduction

This chapter presents the results of the studies on scalability and performance evaluation of the COMET decoupled approach. The studies were carried out based on the simulation models of the system focusing on performance evaluation of content resolution and content delivery processes. The models were defined from the deep analysis of the tasks performed by each COMET entity and the messaging required for performing the involved processes (see details in D3.3 [5] and D4.3 [6]). The input data for our models, corresponding to execution times of the tasks performed in particular entities, were obtained from the experiments carried out on the COMET prototype deployed over the federated testbed, as reported in D6.1 [8] and D6.2 [9].

4.2 Content resolution process

The content resolution process is responsible for discovering all accessible content sources and selecting the best one of them for delivery of the content to the consumer. The selection decision is based on the collected information about the load of content servers' and quality of available routing paths. Once both the server and the path have been chosen, the final step is the configuration of the content delivery path at the server side CAFE (Content Aware forwarding Entity). The last action is not required for content delivered in the best effort service.

Figure 2 shows the simplified scenario of the content resolution process with the involved COMET entities (see D2.2 [1] or D2.3 [2] for details).

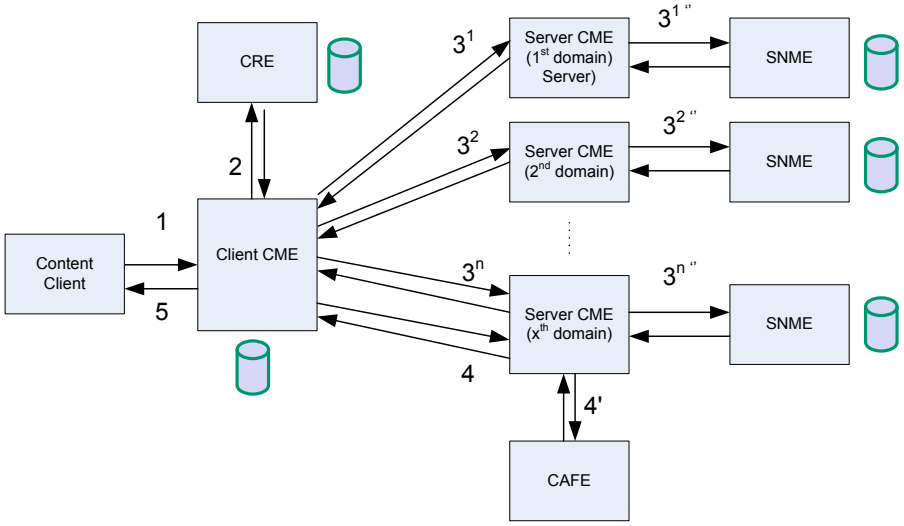


Figure 2: The simplified content resolution process.

For performing this process, the following COMET entities are involved:

- **Content Client** (CC), running at consumer terminal, is responsible for generating content requests and launching the application adequate for the received response, e.g. video player or p2p client;
- **Client Content Mediation Entity** (CME), located in domains with consumers, is responsible for handling content requests, collecting information about the content servers' load and quality of routing paths, selecting preferred content server and invoking content delivery path configuration;

- **Server CME,** located in domains with content servers, provides information about available routing paths and content servers' load, and performs content delivery path configuration at the server side;
- **Server and Network Monitoring Entity** (SNME), located in the server domain, collects information about the load of servers and edge CAFEs.
- **Content Aware Forwarding Entity** (CAFE), located in domains, forwards packets from the content server towards the consumer along the chosen content delivery path;
- **Content Resolution Entity** (CRE), located in some domains, keeps track of the content and resolves content names whenever a client CME requests them. Replicating the typical DNS hierarchy, there are two types of CREs: (1) the authoritative CREs, which store metadata of the content, and (2) the root CREs, which map domains with the authoritative CREs.

In the content resolution process, we distinguish five steps that are executed one after the other:

Step 1: CC sends the content request with content name to the client CME.

Step 2: Client CME asks the CRE (first the root CRE and next the authoritative CRE) for Content Record (CR), which contains information about the content and localisation of accessible content sources.

Step 3: After receiving information from CREs, the client CME asks the indicated server CMEs about the servers' status of accessible content sources and routing paths.

Step 4: client CME asks the chosen server CME to configure the server side CAFE. (This step is not needed for content delivered in best effort service).

Step 5: client CME responds to CC with the URL of the content.

Figure 3 shows the task sequence executed in consecutive way by different COMET entities in order to handle the content request emitted by CC. The assumed notation for describing a task is a pair (y, z), where y is the number of consecutive task (y=1, …, 13) and z denotes the COMET entity, where this task is executed. The task descriptions are presented in the following sections. Note that sequence of tasks 6, 7 and 8 is executed in parallel by server CME responsible for a given content source (from 1 to n).

The key performance indicator of content resolution process in COMET decoupled approach is the Content Resolution Time (CRT) metric proposed in D5.1 [7]. The CRT is defined as time interval starting when a CC sends the content request to the client CME and finishing when the CC receives the URL of the content server. The CRT is the sum of times needed to perform defined above resolution steps from 1 to 5. As stated in D5.1 [7], we assume that the execution time of the content resolution process satisfies consumers when the value of CRT for 95 per cent of content requests does not exceed 2.5s.

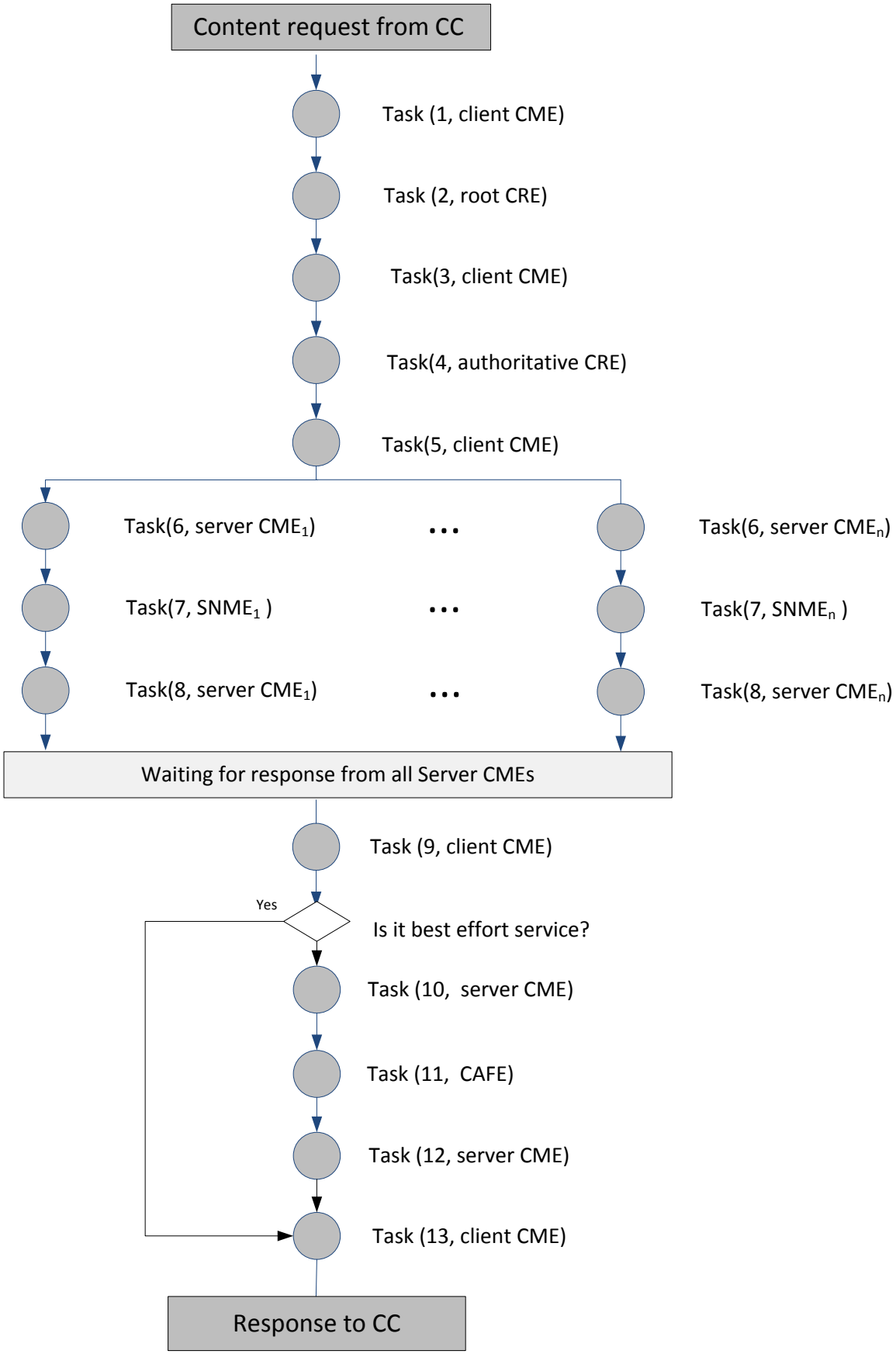


Figure 3: The content resolution process described as a sequence of tasks executed in particular COMET entities to handle content request.

4.2.1 The simulation model

In this section, we briefly describe the simulation model used for performance evaluation of the content resolution process.

From a queuing theory point of view, the system under examination belongs to the *open queuing network* [34], [35], where particular COMET entities play the role of different servers. The system is fed by external arrivals requiring service in such queuing network. The service of an arrival covers the number of tasks that are executed in a predefined sequence. Furthermore, the execution of a given task should be performed by a predefined server.

We construct the simulation model derived from the point of view of handling content requests coming from CCs of a given domain. The service of such requests is performed by the client CME, which controls the resolution process. In order to achieve this , the CME sends requests to other COMET entities (see Figure 2) to get necessary information or to enforce configuration. Since the client CME plays the key role in handling the resolution process, we model its behaviour as closely as possible while the rest of the involved COMET entities are modelled in a simplified way.

We assume that content requests generated by customers arrive to the client CME according to the Poissonian process with the rate □. We argue that the assumption about Poissonian arrival is justified in this case because the requests are generated by a large population of consumers, each being independent of others. In addition, we assume that the correlation level between the COMET entities is rather low, so we assume independence between them.

4.2.1.1 Client CME

The simulation model of the client CME is quite complex since this CME controls the execution of the entire content resolution process. In particular, in the client CME the following tasks are executed (see Figure 3):

- **Task 1** is related to processing of requests received from CCs and sending request to the root CRE. The execution time of task 1 is denoted as T.1.
- **Task 3** covers processing the answer from the root CRE and sends request to the authoritative CRE. The execution of task 2 is denoted as T.2.
- **Task 5** is related to sending requests to server side CMEs asking about the server and path status. The number of messages depends on the number of accessible content sources, which are included in CR. The execution time of task 5 depends on the number of domains n (n=1, 2, 3,...) with accessible content sources.
- **Task 9** is processing the answers from the server CMEs and sending request for path configuration to the chosen server CME. The execution time of task 9 is denoted as T.9.
- **Task 13** is related to processing of the answer from the server CME related to path configuration and sending response to the CC. The execution time of task 13 is denoted as T.13.

The above listed tasks are executed in a consecutive way. The delay is caused by waiting and execution times of the task and the time to communicate other COMET entities.

The queuing model of the client CME is the single server with FIFO queue with deterministic feedback and delay. This means that in this model, we take into account the sequence of tasks that are executed one after the other but after execution of a task, the next task belonging to this sequence enters again the system not immediately but after some time. Moreover, we assume infinite buffer size to avoid losses in the system.

Figure 4 shows the queuing model of the client CME. The client CME is fed by the arrivals of the sequences as depicted in Figure 5. The sequence#1 represents the foreground traffic for which we collect the statistics related to CRT starting from the arrival of the first task in the sequence and finishing when the last task of the sequence ends its execution. The process describing the arrivals of the first tasks is assumed to follow Poissonian distribution with rate □1. The simulation is

performed for the sequence of tasks therefore the collected statistics reflect the impact of correlations between waiting times of particular tasks executed in the client CME. The handling process of the sequence #1 in the system is the following. When task 1 arrives to the system, it joins the FIFO queue for its service. After the service of task 1 has been finished, the message is sent outside the client CME to the root CRE. When the response arrives, task 3 is activated and it joins the FIFO queue. The process is continued until the service of the last task in the sequence is finished.

Since in our experiments we want to stress the domain client CME with high load, we are not able to run simulations with sequence#1 only. This is caused by the fact that simulation experiments in this case are time consuming. Therefore, we decided to limit the foreground traffic by introducing background traffic corresponding to the executions of the sequences #2, #3, #4 and #5, all of them generated with rate □2 and being the sub-sequences of the sequence #1. According to this we will simulate the client CME fed by the sequence of tasks with the rate □ = □1 +□2. The arrivals of the first tasks belonging to the sequences #2, #3, #4 and #5 are also assumed to follow Poissson distribution.

For specifying the tasks in a sequence, we use notation (x, y, z), where x is the number of the sequence, y is the number of the task that follows the numbering of tasks from the content resolution process (x=1,...,13) and z denotes the COMET entity, where the task is executed.

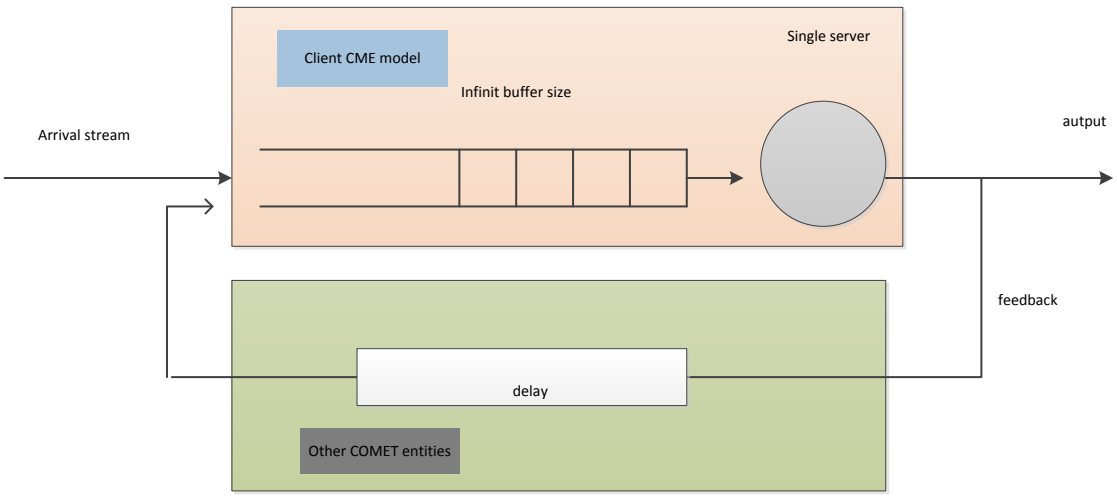


Figure 4: Simulation model of the client CME: single server with FIFO queue and deterministic feedback with delays.

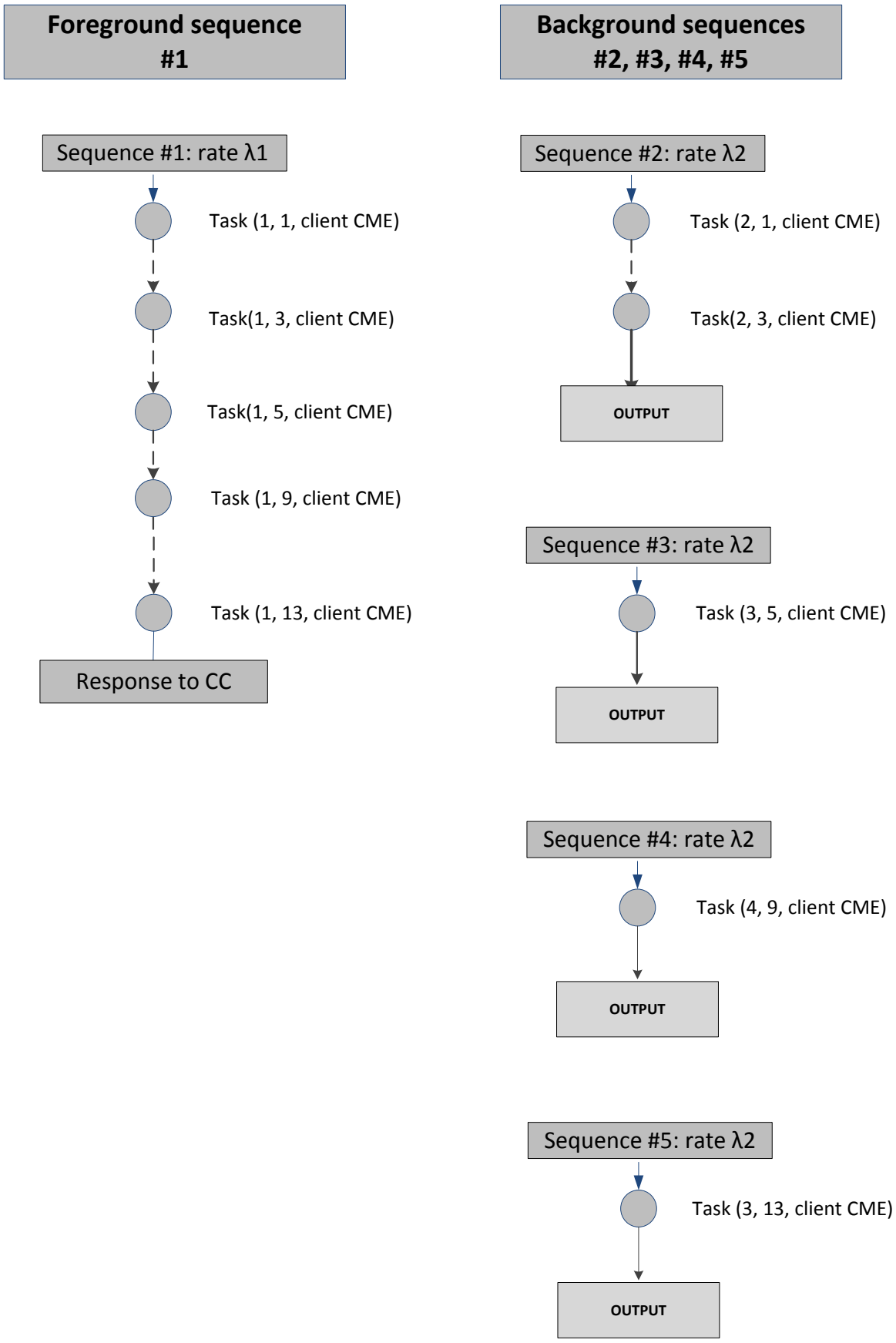


Figure 5: The sequences of the tasks served in the client CME: Sequence #1 is the foreground traffic and the sequences #2, #3, #4 and #5 are the background traffic.

4.2.1.2 Root CRE

The simulation model of the root CRE is quite simple since it performs only one task (see Figure 3):

- **Task 2** is related to the name resolution process and finding the authoritative CRE. The execution time of task 2 is denoted as T.2.

The model of the root CRE is simply the system with single server and FIFO queue, presented in Figure 6.

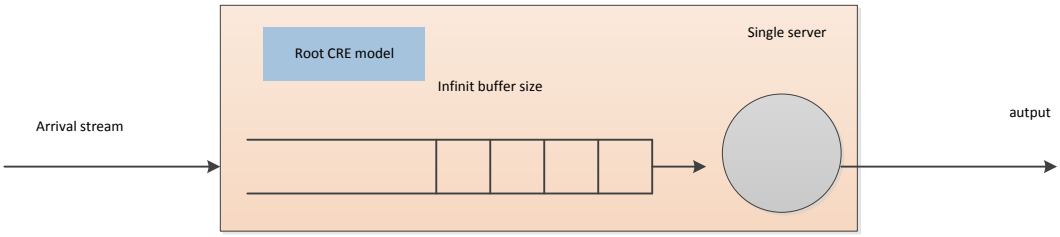


Figure 6: Simulation model of the root CRE; single server with FIFO queue.

This system is fed by three sequences (see Figure 7): sequences #1.1 and #2.1 as mentioned in Figure 5 with cumulative rate □1+□2, and sequence #6 representing requests generated from other client CMEs in the COMET system. The first tasks belonging to the sequence #6 arrive to the system with rate □6. Again, the foreground traffic is generated by the arrivals of the sequence #1.1. In our simulation, we assume that □1 << □2+ □6.

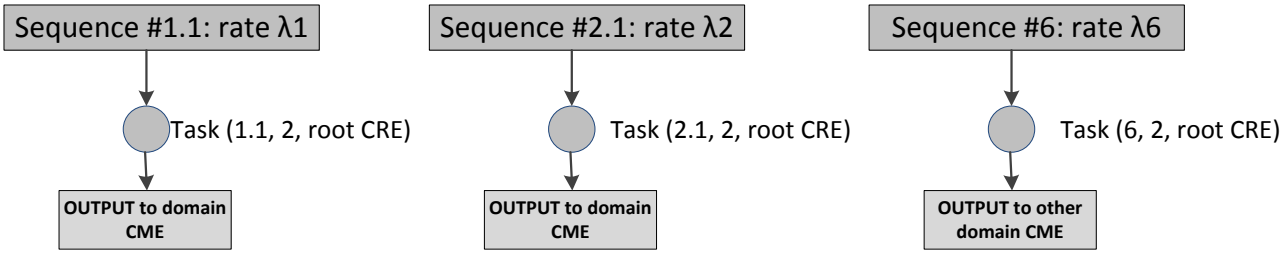


Figure 7: The sequences of the tasks served in the root CME: Sequence #1.1 is the foreground traffic and the sequences #2.1 and #6 are the background traffic.

4.2.1.3 Authoritative CRE

The simulation model of the authoritative CRE is quite simple since it performs only one task (see Figure 3):

- **Task 4** is related to name resolution process and finding the Content Record (CR), which stores information about content and accessible content sources. The execution time of task 4 is denoted as T.4.

The model of the authoritative CRE is similar to the model assumed of the root CRE and is simply the system with single server and FIFO queue (see Figure 8).

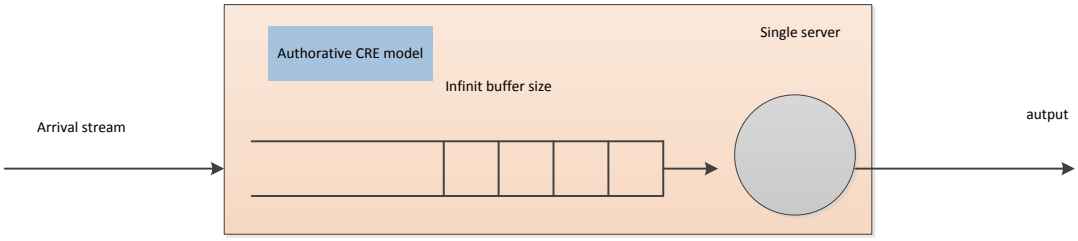


Figure 8: Simulation model of the root CRE; single server with FIFO queue.

This system is fed by two sequences (see Figure 9): sequence #1.3 as mentioned in Figure 5 with cumulative rate □1+□2, and sequence #6 representing the tasks generated from other client CMEs in the COMET system. Again, the foreground traffic is generated by the arrivals of the sequence #1.3 and □1<<□7.

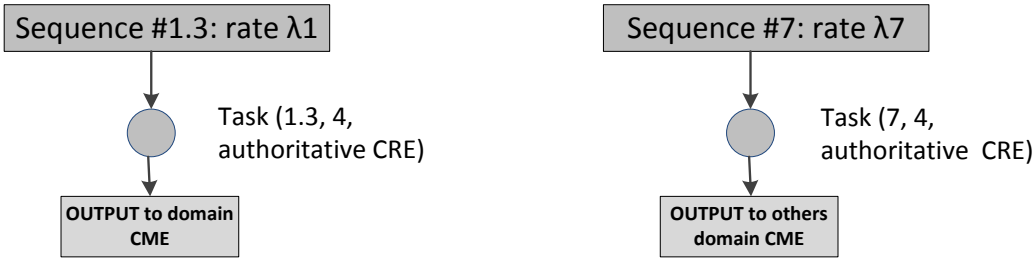


Figure 9: The sequences of the tasks served in the root CRE: Sequence #1.3 is the foreground traffic and the sequence #6 is the background traffic.

4.2.1.4 Server CME, SNME and CAFE

The server CME, SNME and CAFE are modelled together as the queuing network with 3 servers depicted in Figure 10.

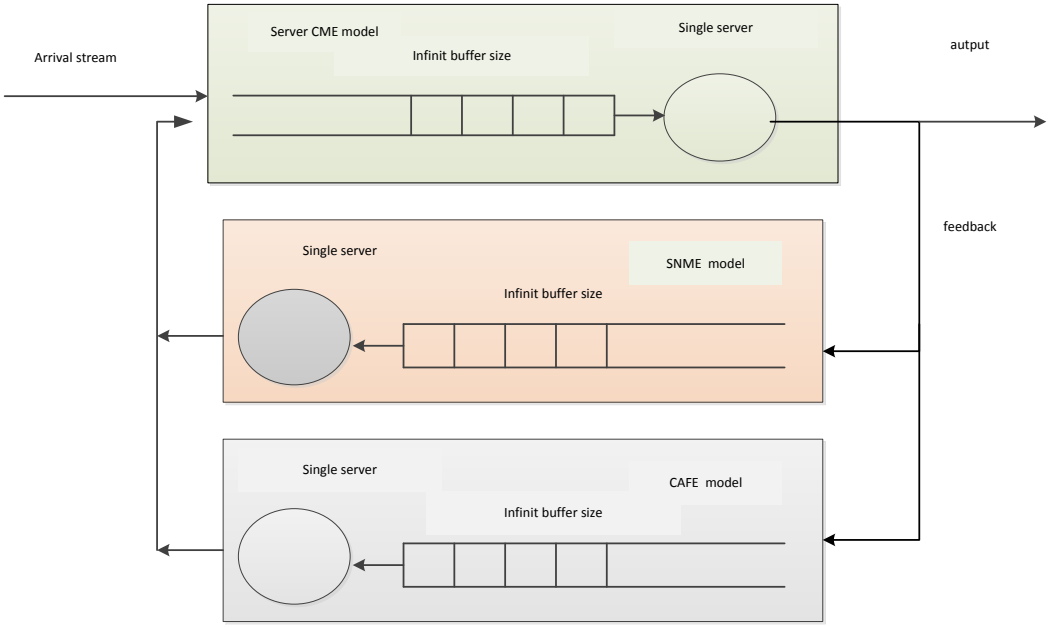


Figure 10: The model of server CME: single server with FIFO discipline and deterministic feedback with delays.

In the server CME four tasks are executed.

1. **Task 6** corresponds to processing server status requests and sending request to SNME. The execution time of task 6 is denoted as T.6.

2. **Task 8** corresponds to processing answer from SNME and answering to the client CME. The execution time of task 8 is denoted as T.8.
3. **Task 10** corresponds to processing Path configuration request received from the client CME and sending request to CAFE. The execution time of task 10 is denoted as T.10.
4. **Task 12** corresponds to processing the CAFE response and sending the answer to the client CME. The execution time of task 12 is denoted as T.12.

In the SNME server, only one task is executed:

1. **Task 7** corresponding to processing server status request and answering to the server CME. The execution time of task 7 is denoted as T.7.

In the CAFE server, only one task is executed:

1. **Task 11** corresponding to the configuration of the content delivery path at CAFE. The execution time of task 11 is denoted as T.11.

The discussed model is simulated before the main simulation. The idea is to collect delay statistics. When we run the main simulation with the foreground traffic (sequence #1), we simply treat this sequence as the tested sequence which enters the server CME. In this case, the tasks of the foreground sequence are the tasks indicated in the sequences #8 and #9, respectively.

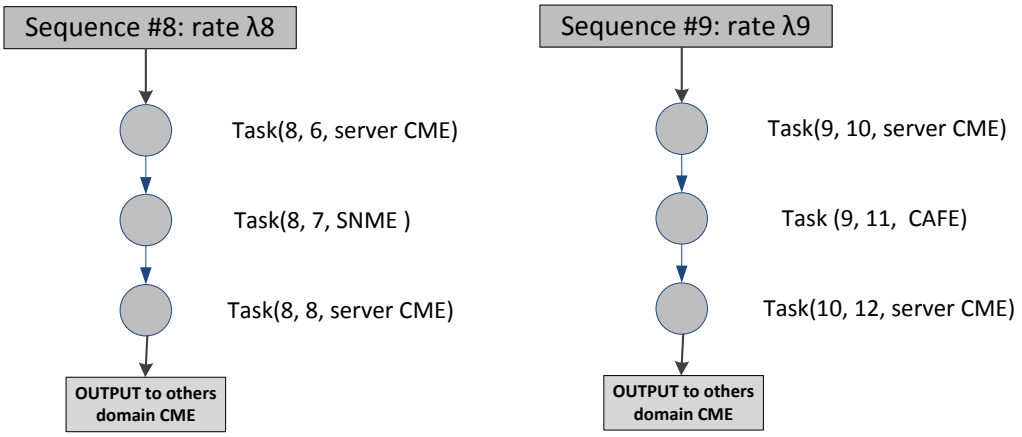


Figure 11: The sequences of tasks submitted to the common model for server CME, SNME and CAFE.

4.2.2 Evaluation of the basic COMET system

In this section, we present numerical results showing the effectiveness of content resolution in the basic COMET system, i.e., we do not consider caching in client CME and we assume a basic algorithm for querying CMEs at the server side for executing the tasks 6, 7 and 8.

The numerical results are obtained from the simulation model described in previous section.

4.2.2.1 Values of the input parameters for simulation

In order to run the simulations, we need to fix the values of the parameters describing the simulation model. More specifically, we fix: (1) values of the service times of particular tasks executed in the COMET entities, (2) the arrival rates of the simulated sequences and (3) the network model. Table 2 shows the default values of the parameters assumed in our simulation.

The network model was presented in chapter 3.2. The model includes the network topology for getting information about the "distances" between the engaged COMET entities (represented by the Round Trip Time - RTT), the content characteristics and server locations, and the distribution of content replicas.

Table 2: Default values of parameters describing simulation model.

<table>
<tr><td colspan="10">Number of content sources for the most popular content</td></tr>
<tr><td colspan="10">100</td></tr>
<tr><td colspan="10">RTT delays in the network</td></tr>
<tr><td colspan="2">AS tier 1</td><td colspan="2">AS tier 2</td><td colspan="2">AS tier 3</td><td colspan="2">clientCME - rootCRE</td><td colspan="2">clientCME – Auth.CRE</td></tr>
<tr><td colspan="2">61.3ms</td><td colspan="2">21.5ms</td><td colspan="2">1.5ms</td><td colspan="2">40ms</td><td colspan="2">10ms</td></tr>
<tr><td colspan="10">client CME</td></tr>
<tr><td colspan="8">Service times</td><td colspan="2" rowspan="2">Load</td></tr>
<tr><td>T1</td><td colspan="2">T3</td><td colspan="4">T5</td><td>T9</td><td>T13</td></tr>
<tr><td>5ms</td><td colspan="2">5ms</td><td colspan="4">5ms
+0.5ms for sending of an additional inquiry
+0.25ms for looking up an additional record in cache</td><td>5ms</td><td>5ms</td><td>0.7</td></tr>
<tr><td colspan="10">server CME</td></tr>
<tr><td colspan="8">Service times</td><td colspan="2" rowspan="2">Load</td></tr>
<tr><td colspan="4">T6, T8</td><td colspan="4">T10, T12</td></tr>
<tr><td colspan="4">2ms</td><td colspan="4">3ms</td><td colspan="2">0.7</td></tr>
<tr><td colspan="5">root CRE</td><td colspan="5">authoritative CRE</td></tr>
<tr><td colspan="3">Service time T2</td><td colspan="2">Load</td><td colspan="3">Service time T4</td><td colspan="2">Load</td></tr>
<tr><td colspan="3">1ms</td><td colspan="2">0.7</td><td colspan="3">1ms</td><td colspan="2">0.7</td></tr>
<tr><td colspan="5">SNME</td><td colspan="5">CAFE</td></tr>
<tr><td colspan="3">Service time T7</td><td colspan="2">Load</td><td colspan="3">Service time T11</td><td colspan="2">Load</td></tr>
<tr><td colspan="3">5ms</td><td colspan="2">0.7</td><td colspan="3">25ms</td><td colspan="2">0.7</td></tr>
</table>

Let us recall that the values of the service times of the particular tasks were obtained from experiments carried out in the COMET federated testbed. The description of the experiment is provided in D6.2 [9]. We assume that service times of particular tasks are constant.

4.2.2.2 Case 1: Simulation results for default load system

The results of the simulations are: (1) the statistics of the Content Resolution Time (CRT), where μ denotes the average value, σ denotes standard deviation and q(0,95) is 95 percentile, and (2) the execution delay of particular resolution sub-processes and resolution steps from 2 to 5. These delays are measured for:

- Name Resolution sub-process (NR), performed by root and authoritative CREs. The NR delay corresponds to waiting and execution times of tasks T2 and T4;
- Path and Load Retrieval sub-process (PLR), performed by server CME and SNME. The PLR delay corresponds to waiting and execution times of tasks T6, T7 and T8;
- Path Configuration sub-process (PC), performed by server CME and CAFE. The PC delay corresponds to waiting and execution times of tasks T10, T11 and T12;
- Step2, covering the waiting and execution times of tasks 1, 2, 3, and 4.
- Step3, covering the waiting and execution times of tasks 3, 6, 7, and 8.
- Step4, covering waiting and execution times of tasks 9, 10, 11 and 12.
- Step5, covering waiting and execution time of task 13.

The simulation results are presented in Table 3. We assumed that the total request rate of background sequences arriving to client CME (sequences #2, #3, #4 and #5), is □2 = 23.93 requests/s, which offers the load in client CME equal to 0.7. The rate of foreground sequence (#1) is negligible because □1 equals 1/150 of □2. The request rates of background sequences (#6, #7,

#8 and #9) arriving to root CRE, authoritative CRE and server CME was fixed to produce load in these entities equal to 0.7. Note that both sequences #8 and #9 load the server CME.

Table 3: Simulation results for the case of the basic system with default parameters.

Total delay [s] CRT			Average delays in sub processes [s]			Average delay in client CME
μ	σ	q(0.95)	NR	PLR	PC	
0.515 ± 0	0.070 ± 0.001	0.644 ± 0.001	0.053 ± 0	0.166 ± 0.001	0.214 ± 0	0.081 ± 0.001
Average delays in steps 2-5 [s]						
Step 2		Step 3		Step 4		Step 5
0,085 ±0,001		0,185 ±0		0,230± 0,001		0,015 ± 0,001

The results from Table 3 show that the mean CRT is about 0.5s and the CRT does not exceed the value 0.65s for 95 % of the content requests under system load equal to 0.7. Let us recall that the system meets user expectations since the 95 percentile of CRT is lower than 2.5 s. By comparing the execution times of the resolution steps, we can observe that the most time consuming steps are 3 and 4. Both of them are executed in the server CME so the essential part of the delay is due to the distance between the client CME and the server CME. Therefore, we can conclude that the most significant delay in the COMET resolution process comes from the communication between COMET entities located in different domains.

For comparison purposes, in Table 4 we present the discussed delays obtained for the case where consumer and content server are located in the same domain.

Table 4: Simulation results for the case where consumer and content server are located in the same domain.

Total delay [s] CRT			Average delays in sub processes [s]			Average delay in client CME
μ	σ	q(0.95)	NR	PLR	PC	
0.231 ± 0.001	0.070 ± 0.001	0.367 ± 0.003	0.054 ± 0.001	0.018 ± 0	0.075 ± 0.001	0.084 ± 0.001
Average delays in steps 2-5 [s]						
Step 2		Step 3		Step 4		Step 5
0.085 ± 0		0.038 ± 0,001		0.092 ± 0,001		0.016 ± 0.001

As expected, the CRT delay decreases compared to the case when the client and the server were located in different domains. The obtained CRT delays are in a similar range as the ones measured for CDN network. The CDN results reported in [37] are about 0.130 s, while in the decoupled approach we have 0.23 s. We can conclude that such difference is not noticeable by the consumers. Note that we have higher CRT values in the COMET system because of the fact that we execute more tasks and some of them are performed outside the client domain.

4.2.2.3 Case 2: Simulation results for the basic system under increased load

This section presents the simulation results for the basic system (the COMET entities are located in different domains) when the content request rate increases compared to case 1. We distinguish between the situation when the request rate increases only in the client domain and the situation when the request rate increases in other domains.

Table 5 shows the simulation results when the client CME load is 0.7 and 0.8 while the load of other COMET entities (root and authoritative CRE, server CME) is 0.7 or 0.8.

Table 5: Simulation results for the case of the basic system under increasing load.

Load		Total delay [s] CRT			Average delays in sub processes [s]			Average delay in client CME
client CME	Other entities	μ	σ	q(0.95)	NR	PLR	PC	
0.7	0.7	0.515 ± 0	0.070 ± 0.001	0.644 ± 0.001	0.053 ± 0	0.166 ± 0.001	0.214 ± 0	0.081 ± 0.001
0.8	0.7	0.553 ± 0.001	0.100 ± 0.002	0.740 ± 0.003	0.053 ± 0	0.165 ± 0.001	0.214 ± 0	0.120 ± 0.001
0.7	0.8	0.548 ± 0.001	0.083 ± 0.001	0.703 ± 0.002	0.056 ± 0	0.173 ± 0	0.237 ± 0.001	0.081 ± 0.001
0.8	0.8	0.585 ± 0.002	0.108 ± 0.002	0.791 ± 0.004	0.056 ± 0	0.173 ± 0	0.237 ± 0	0.120 ± 0.002

The results indicate that the CRT delay is not sensitive when the load increases from 0.7 to 0.8. Under these relatively high loads, we do not observe critical entities in the system, which could fall to the congestion state (and produce large delays) easier than others do. Therefore, we can admit a load until 0.8 without any essential degradation of the service.

4.2.2.4 Case 3: Simulation results for the basic system under increased number of content sources

This section presents the simulation results for the increased number of content sources from 100 up to 1000. For other contents, the number of content sources is scaled following popularity factor defined by zipf distribution. Notice, that the execution time of task 5 (see Table 2) is proportional to the number of content sources. In addition, the activation of task 9 waits for all responses coming from the inquired server CMEs.

Table 6 shows the simulation results for the basic system with a number of content sources equal to 100 and 1000. One can observe that in the case when the number of content sources increases, we have essentially more queries to server CMEs (in average, about 6 times more). As a consequence, the CRT also increases, in the studied case about 3 times in q(0.95) parameter. This effect is rather not desirable, therefore we have proposed Smart Querying Algorithm (SQA) to overcome it. The effectiveness of this algorithm is studied in section 4.2.3.

Table 6: Simulation results for the case of the basic system under increased number of content sources.

No. content sources	□2 [1/s] in client domain	Total delay [s] CRT			Average delays in sub processes [s]			Average delay in client CME [s]	Average duration of task T5 [s]	Average number of sent requests in T5
		μ	σ	q(0.95)	NR	PLR	PC			
100	23.93	0.515 ± 0	0.070 ± 0.001	0.644 ± 0.001	0.053 ± 0	0.166 ± 0.001	0.214 ± 0	0.081 ± 0.001	0.009 ± 0	9.43 ± 0.03
1000	12.81	0.817 ± 0.006	0.390 ± 0.011	1.592 ± 0.020	0.053 ± 0	0.173 ± 0	0.214 ± 0	0.375 ± 0.006	0.034 ± 0	59.75 ± 0.35

4.2.2.5 Case 4: Simulation results for the basic system under increasing domain sizes

In this case, we study the situation when particular domains grow up and, consequently, the distances (measured by RTT) between COMET entities increase. For this purpose, we increase the RTT delays as indicated in Table 7.

Table 7: Exemplary values of RTT assumed for default and larger domain sizes.

Analysed system	RTT delays in the network					
	AS tier 1 [ms]	AS tier 2 [ms]	AS tier 3 [ms]	client CME – root CRE [ms]	client CME – Auth.CRE [ms]	server CME – CAFE [ms]
Default size	61.3	21.5	1.5	40	10	10
Larger size	94.3	36.9	6.2	60	17	17

The simulation results are presented in Table 8. One can observe that if we increase the domain sizes then the CRT delay also increases. For instance, the mean value of the CRT is greater of 0.243 s and is very close to increased value of the q(0.95) parameter. This is because the RTT values in our simulations are assumed to be constant. Concluding, we highlight that the RTT values have great impact on the COMET system performances. So, the proper location of COMET entities in the network is important. The problem is that, in general, we have fewer possibilities for choosing the places in the network to deploy COMET system. Instead, we may decide about which information should be stored in local caches.

Table 8: Comparison of simulation results for basic system with default parameters and increased domain sizes.

Analysed system	Total delay [s] CRT			Average delays in sub processes [s]			Average delay in client CME [s]
	μ	σ	q(0.95)	NR	PLR	PC	
Default size	0.515 ± 0	0.070 ± 0.001	0.644 ± 0.001	0.053 ± 0	0.166 ± 0.001	0.214 ± 0	0.081 ± 0.001
Larger size	0.758 ± 0.001	0.080 ± 0.001	0.898 ± 0.001	0.081 ± 0	0.273 ± 0.001	0.323 ± 0.001	0.081 ± 0.001

4.2.2.6 Conclusions

The reported results showing effectiveness of the content resolution process as performed by the basic COMET system indicate that:

- The CRT delays meet the requirements even for relatively highly loaded COMET entities (0.8)
- There is no critical entity in the COMET system, which falls to the congestion state easier than others do.
- It was recognised that the retrieval of server loads and paths information might become the critical procedure (which essentially extend the CRT delay) for the case of the large number of content sources.
- The RTTs between communicating COMET entities have significant impact on the CRT delays.

4.2.3 Evaluation of the extended COMET system

The simulation results presented in the previous section indicate that CRT strongly depends on: (1) the communication between COMET entities located far from the client CME or (2) the Path and Load Retrieval sub-process, which becomes time consuming when we handle the request for popular content with many content sources. In the following sections, we study two mechanisms for improving the content resolution process.

4.2.3.1 Case 5: Reducing CRT by using cache mechanisms

In order to minimize transfer of messages between the client CME and other COMET entities, we investigate the cache mechanism, which allows re-using previously retrieved information from the local storage. Figure 12 shows the modified content resolution process comparing to the basic one presented in Figure 3, when we explore the cache mechanism. We can get profit from the cache when:

- The content record related with requested content is stored in the so-called Content Record Cache (CR-Cache). If the content record is available in CR-cache, then, after execution of task 1 we jump directly to task 5. Therefore, there is no communication between the client CME and root CRE and authoritative CRE.
- The address of authoritative CRE is stored in the so-called Authoritative Cache (A-Cache). If the address is available in the A-Cache, then, after execution of task 1 we jump directly to task 3. Therefore, no communication between client CME and root CRE is needed.
- The information about servers load and paths is stored in the so-called Path Cache (P-Cache) and Server Load Cache (SL-Cache). If both information about paths and server load are available in the caches, then, after execution of task 5 we jump directly to task 9. Therefore, we can avoid no communication between client CME and server CMEs.

Let us remark that the information stored in the caches can become out of date. To minimize the risk, we propose to use time-based cache management algorithm with the time-out corresponding to the validity of information.

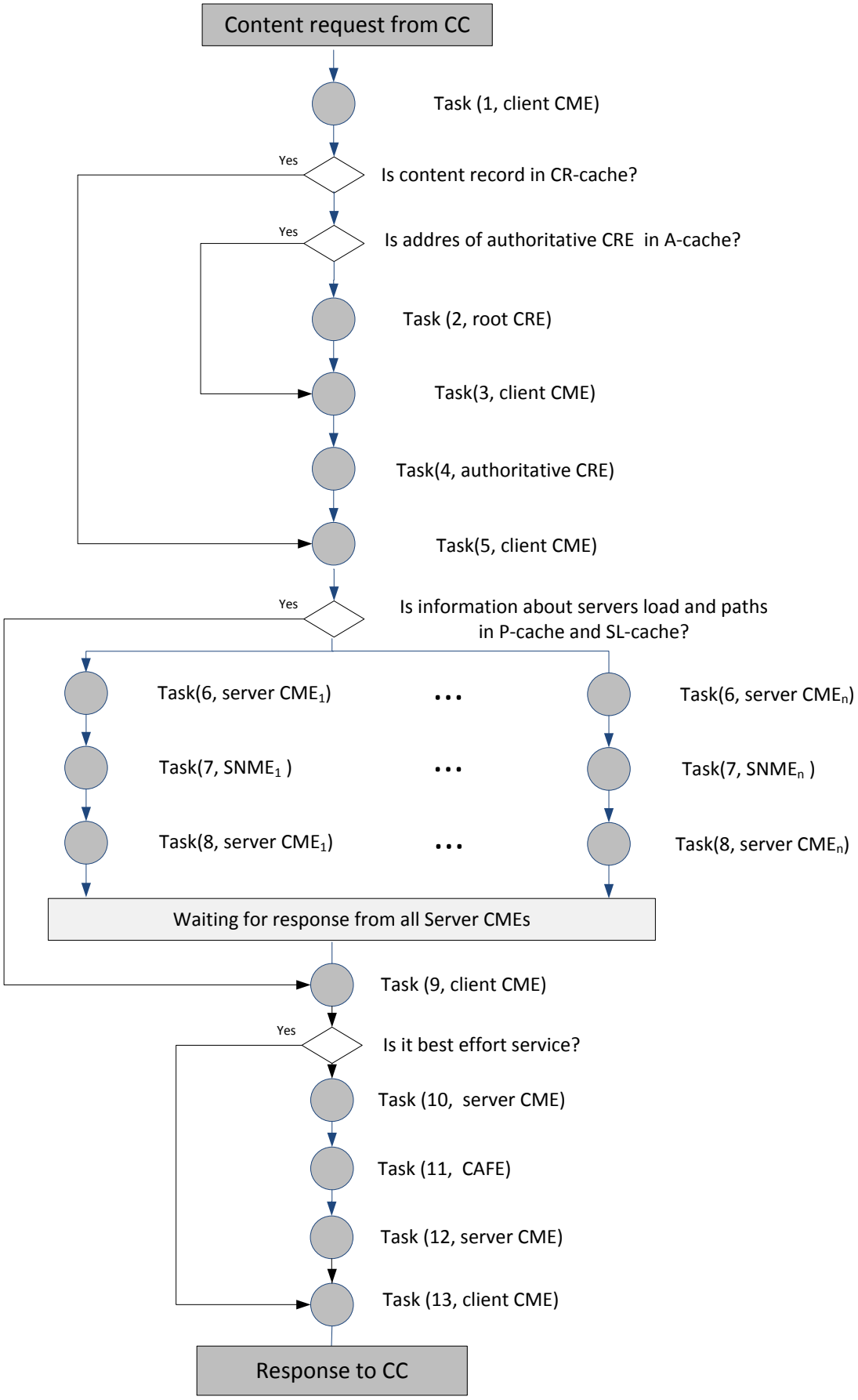


Figure 12: Content resolution process supported by the information stored in the client CME cache.

In Table 9, we present the parameters assumed for proposed caches: CR-Cache, A-Cache, P-Cache and SL-Cache.

Table 9: The parameters assumed for caches.

Cache parameters	CR-Cache	A-Cache	P-Cache	SL-Cache
Size (in records)	$5*10^3$	$5*10^3$	10^4	$3*10^4$
Timeout [min]	30	30	60	15

The simulation results for CRT and the cache hit ratio of all caches used in client CME are reported in Table 10 and Table 11, respectively. As one can observe, the cache hit ratio in the considered system is high. As it was expected, by using cache we are able to essentially reduce the times of sub-processes execution, and, as a consequence, to reduce the CRT delays.

Table 10: Comparison of delays for basic and extended COMET system.

Analysed system		Total delay [s] CRT			Average delays in sub processes [s]			Average delay in client CME [s]
		μ	σ	q(0.95)	*NR*	*PLR*	*PC*	
no cache	Load 0.7	0.515 ± 0	0.070 ± 0.001	0.644 ± 0.001	0.053 ± 0	0.166 ± 0.001	0.214 ± 0	0.081 ± 0.001
	Load 0.8	0.585 ± 0.002	0.108 ± 0.002	0.791 ± 0.004	0.056 ± 0	0.173 ± 0	0.237 ± 0	0.120 ± 0.002
	Load 0.7, increased RTT	0.758 ± 0.001	0.080 ± 0.001	0.898 ± 0.001	0.081 ± 0	0.273 ± 0.001	0.323 ± 0.001	0.081 ± 0.001
with cache	Load 0.7	0.381 ± 0	0.101 ± 0.001	0.543 ± 0.002	0.0017 ± 0	0.101 ± 0	0.214 ± 0	0.071 ± 0.001
	Load 0.8	0.440 ± 0.001	0.131 ± 0.001	0.665 ± 0.003	0.0017 ± 0	0.105 ± 0	0.237 ± 0	0.102 ± 0.001
	Load 0.7, increased RTT	0.552 ± 0.001	0.147 ± 0.001	0.770 ± 0.002	0.0026 ± 0	0.166 ± 0	0.323 ± 0	0.070 ± 0

Table 11: Cache hit ratio.

Analysed system		Per cent of requests served by:				Percentage of requests served entirely by cache memories
		CR-Cache	A-Cache	P-Cache	SL-Cache	
with cache	Load 0.7	96.77% ± 0.02%	96.77% ± 0.02%	88.16% ± 0.25%	39.42% ± 0.25%	39.41% ± 0.25%
	Load 0.8	96.75% ± 0.06%	96.75% ± 0.06%	87.68% ± 0.12%	39.46% ± 0.30%	39.45% ± 0.30%
	Load 0.7,increased RTT	96.75% ± 0.14%	96.75% ± 0.14%	87.98% ± 0.32%	39.34% ± 0.65%	39.33% ± 0.65%

In Table 12 we present the utilisation levels of the involved caches. By utilization level we mean the fill ratio of cache memory (counted in records). Notice, that the fill ratio for the caches with large values of timeouts, e.g. for CR-Cache fixed on 60 min, is about 98% while for the caches with low values of timeouts, e.g. in SL-Cache fixed on 15 min, is about 74%.

Table 12: Fill ratio for used cache memory.

Analysed system		Average cache size (compared to max size):			
		CR-Cache	A-Cache	P-Cache	SL-Cache
with cache	Load 0.7	98.21% ±0.02%	98.21% ±0.02%	99.42% ±0.02%	72.96% ±0.02%
	Load 0.8	98.43% ±0.01%	98.43% ±0.01%	99.49% ±0.01%	76.18% ±0.01%
	Load 0.7, increased RTT	98.20% ±0	98.20% ±0	99.42% ±0.01%	72.97% ±0.01%

4.2.3.2 Case 6: Smart querying algorithm

As it was recognized in case 3, when we assume large number of the content sources for a given content, the execution time of Task 5 may radically increase. This is due to a need to send queries to each indicated server CME and to wait for all responses. Therefore, if at least one of the indicated server CME is far away from the server CME, this time lasts quite long. In order to mitigate this problem, we propose to send the queries to particular server CMEs ordered from the server CME with the smallest RTT up to the server CME with the largest RTT. After receiving a number of responses, we run the multi criteria decision algorithm to choose the best content source. If its decision is unchanged for a certain number of responses, we stop the algorithm and go to the next task (task 9) in the content resolution procedure.

In Table 13 we show the simulation result illustrating advantages of applying Smart Querying Algorithm (SQA). As it was expected, the profit from using SQA is more visible for the case of popular content with large number of content sources.

Table 13: Simulation results showing effectiveness of the SQA algorithm.

Analysed system	Total delay [s] CRT		Average delays in sub processes [s]			Average delay in client CME [s]	Average duration of task T3 [s]	Average number of T3 interruptions:
	μ	q(0.95)	NR	PLR	PC			
Basic, 100 content sources	0.515 ± 0	0.644 ± 0.001	0.053 ± 0	0.166 ± 0.001	0.214 ± 0	0.081 ± 0.001	0.009 ± 0	0 ± 0
SQA, 100 content sources	0.485 ± 0.001	0.617 ± 0.002	0.053 ± 0	0.153 ± 0	0.205 ± 0	0.082 ± 0	0.009 ± 0	0 ± 0
Basic, 1000 content sources	0.817 ± 0.006	1.596 ± 0.020	0.053 ± 0	0.173 ± 0	0.214 ± 0.001	0.375 ± 0.006	0.034 ± 0	0 ± 0
SQA, 1000 content sources	0.547 ± 0.001	0.839 ± 0.003	0.053 ± 0	0.140 ± 0	0.192 ± 0	0.219 ± 0.001	0.021 ± 0	11.42% ± 0.05%

4.2.3.3 Case 7: Cache & smart querying algorithm

In this section we present results for the COMET system, in which we apply caches and SQA algorithm in client CME to reduce CRT delays. Table 14 and Table 15 shows the CRT delays and the cache hit ratio for the system with 100 content sources for the most popular content and the load of COMET entities equal to 0.7. The results show that, the CRT delay is reduced to 0.15s comparing to the basic COMET system (30% less). The obtained CRT delay is the lowest one in all analysed cases.

Table 14: Simulation results for the COMET system with cache and SQA algorithm.

Analysed system	Total delay [s] CRT			Average delays in sub processes [s]			Average delay in client CME
	μ	6	q(0.95)	NR	PLR	PC	
Basic, 100 content sources	0.515 ± 0	0.070 ± 0.001	0.644 ± 0.001	0.053 ± 0	0.166 ± 0.001	0.214 ± 0	0.081 ± 0.001
CR, A, P, SL-Caches and SQA 100 content sources	0.367 ± 0.001	0.105 ± 0	0.534 ± 0.001	0.0017 ± 0.001	0.095 ± 0	0.205 ± 0.001	0.071 ± 0.001

Table 15: Cache hit ratio for COMET system with cache and SQA algorithm.

Analysed system	Per cent of requests served by:				Percentage of requests served entirely by cache memories
	CR-Cache	A-Cache	P-Cache	SL-Cache	
CR, A, P, SL-Caches, SQA and 100 content sources	96.76% ± 0.04%	96.76% ± 0.04%	88.04% ± 0.24%	39.27% ± 0.33%	39.26% ± 0.33%

4.2.3.4 Conclusions

The reported results showing effectiveness of the content resolution process performed by the extended COMET system indicate that:

- Caching mechanisms used in the client CME for storing content/authoritative records, paths and servers' load may essentially reduce CRT delay because the client CME avoids communication with other COMET entities;
- SQA (or similar) algorithm for the retrieval of server loads and paths may reduce the CRT delay especially for the case of large number of content sources.

Concluding, we recommend applying both discussed mechanisms in the COMET system.

4.3 Content delivery process

In this section, we assess scalability of content delivery process performed by CAFEs. From the knowledge gained from the CAFE design, implementation and validation, we identified complexity of packet forwarding and overhead introduced by COMET header as two areas of the content delivery process that may affect COMET scalability.

4.3.1 Performance of packet forwarding

The content delivery process covers operations that are performed in a time scale of microseconds, i.e., forwarding of COMET packets. They suppose a real challenge for the performance of the system as well as crucial point for understanding the potential scalability limitations of COMET system.

The commonly accepted approach to study scalability is to compare the studied system with other benchmark systems that are commonly considered to be scalable [36]. We applied this approach in the analysis of the CAFE performance presented in D6.2 [9]. Based on the obtained results, we concluded that CAFE performance is similar to the performance of an IP router. From the scalability point of view, we analysed CAFE performance by considering the two main operations performed in the CAFEs: interception and forwarding.

Interception is the operation that consists of taking IP packets and encapsulating them with the COMET header. Forwarding is the transfer of COMET packets based on those headers, as one can see in Figure 13.

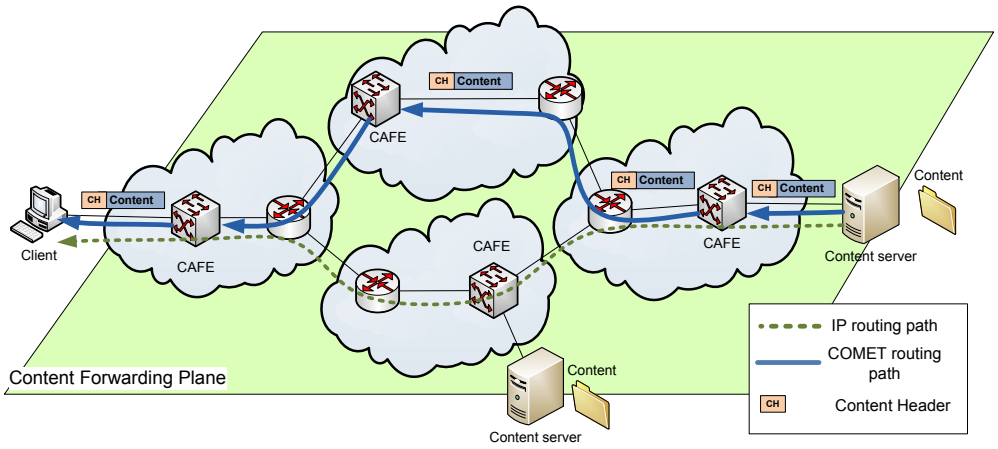


Figure 13: The principle of COMET forwarding process.

Interception is performed in the edge CAFE solely, whereas forwarding is performed in all CAFEs (edge and core) along the content delivery path. As presented in D6.2 [9], interception operation is much more exigent than forwarding from the performance point of view. Therefore, the most demanding operations are performed in the edge CAFE only. This is a good approach for addressing scalability as it was proved in the case of DiffServ architecture [38], where per-flow operations are performed only at the edge routers. In fact, the IntServ architecture [39] was not widely deployed mainly since the operations performed in core routers were related with each single flow. On the contrary, the DiffServ architecture is widely deployed because it eliminates the complex per-flow operations in the backbone, allowing its deployment in large networks. Therefore, we do not see special scalability issues in CAFE performance.

4.3.2 Analysis the length of forwarding key list

The COMET forwarding concept follows the source routing paradigm, where each packet includes information about content delivery path at domain level. We will analyse scalability by comparing the proposed solution for CAFE forwarding with IPv6 forwarding. We will calculate the upper limit and percentiles of the key length and forwarding list length and compare with the overhead in IPv6.

The list of forwarding keys inserted in the COMET header is processed by the consecutive CAFEs along the path. Each CAFE in the path uses one forwarding key from the list in order to forward the packet to appropriate next CAFE. The connectivity between peering CAFEs may be provided by different underlying technologies, e.g. Ethernet, IP tunneling, MPLS, VPLS. The information encrypted in the forwarding key determines the following CAFE on the path (which is the next CAFE hop and the forwarding rule) and the COMET Class of Service required for serving the packet during the transmission see example presented in Figure 14.

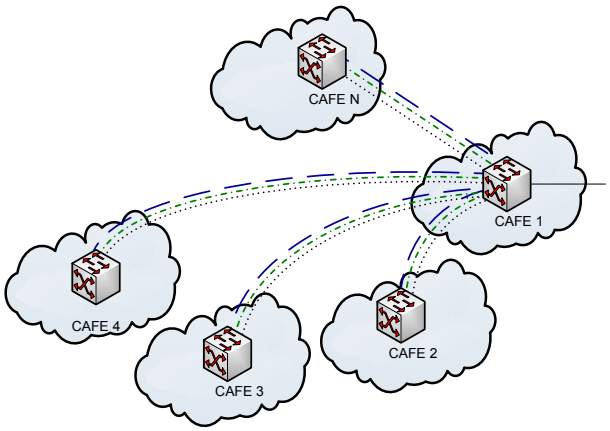


Figure 14: Next CAFE hops in the domain.

The number of different keys required in a given domain is related to domain degree and offered CoS, and can be calculated as in (1).

$$N_{of_{keys}} = \mathrm{N}_{\mathrm{of_{CoS}}} \times \mathrm{degree}_{\mathrm{of_{domain}}} = 3 \times \mathrm{degree}_{\mathrm{of_{domain}}} \quad (1)$$

From the large-scale model presented in chapter 3, we calculate the histogram of domain's degree, which is presented in Figure 15. The model covers 36,878 domains and 206,969 links with an average domain's degree equal to 5.61. The maximum degree of any domain is 2,972 adjacent domains.

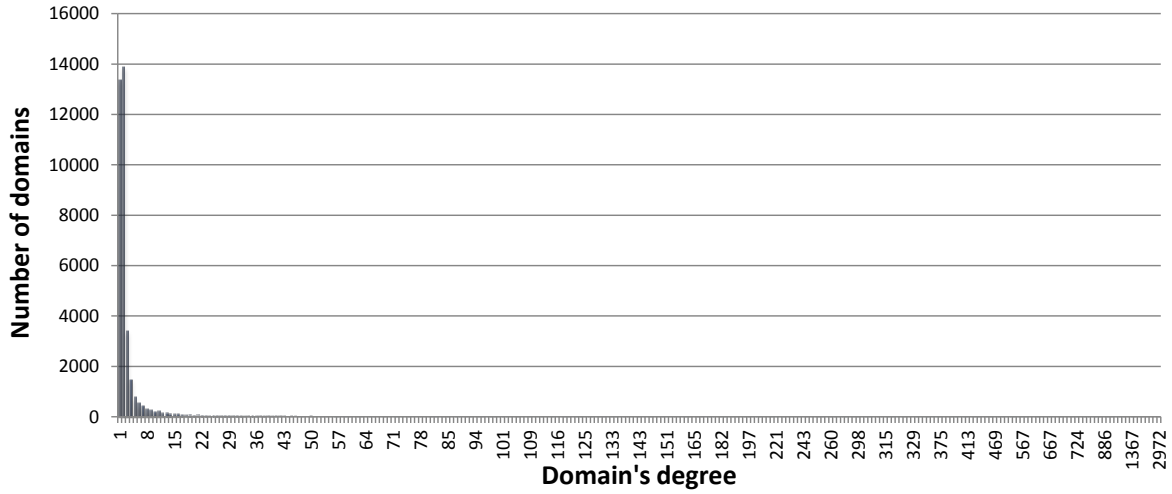


Figure 15: Histogram of domain's degree.

The percentiles of the histogram, together with the necessary number of keys and the necessary key length for each percentile are presented in Table 16.

Table 16: Percentiles of domain's degree histogram and N_of_keys.

Percentile [%]	Degree is less or equal	Number of keys	Key length [bit]
90%	6	18	5
99%	60	180	8
99,50%	128	384	9
99,90%	512	1536	11
99,99%	2000	6000	13
100%	2972	8916	14

These results show that a forwarding key of 1 byte size is enough for most domains (99%). This is the case for all stub, tier 3 and most tier 2 domains. On the other hand, the largest transit domains, like tier 1 and some tier 2 domains will require 2 bytes for forwarding key.

The list of forwarding keys included in the packet header depends on the path length. If we consider that there is one CAFE per domain and considering the number of domains in the current Internet [14], then we can calculate the number of forwarding keys in the list. Based on the large-scale model presented in chapter 3, we calculate the probability density function (pdf) of the path length in the current Internet that is shown in Figure 16. The number of considered domains is 36,878, so the number of paths is bigger than 1.3 10^9. The average length of paths is 4.88 domains, and the maximum length of any path is 20 domains.

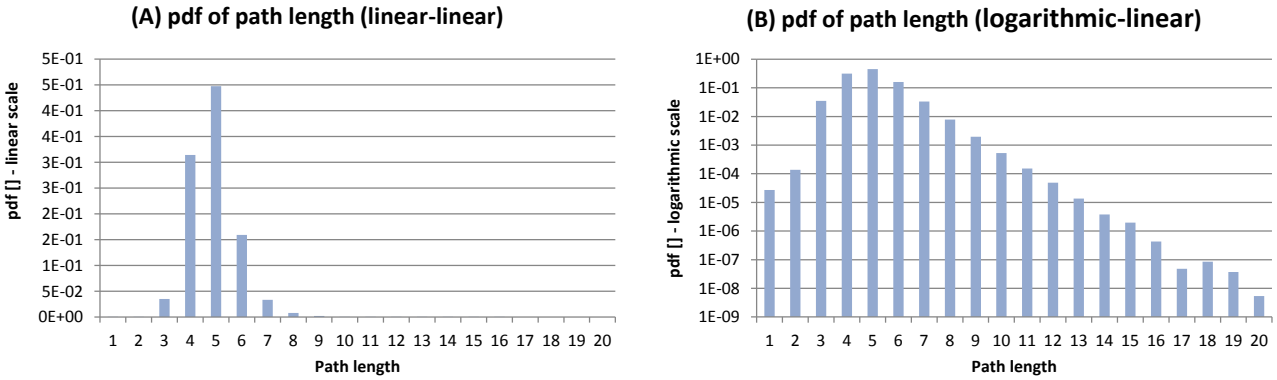


Figure 16: Probability Density Function (pdf) of path length,
where (a) linear scale, (b) logarithmic scale.

In Table 17, we present the percentile of path length distribution to get better understanding of distribution. The results say that most of the paths are shorter or equal to 6 domains.

Table 17: Percentiles of path length in the current Internet.

Percentile [%]	Path length
90%	6
99%	7
99,50%	8
99,90%	9
99,99%	11
100%	20

Concluding, the length of forwarding keys in each packet is less or equal to 8 bytes for the 99% of cases in the current Internet (assuming path length of 7 domains and one additional byte to cross tier 1 domain). On the other hand, in the extreme case, we should consider 21 keys of 2 bytes, i.e., 42 bytes. Therefore, we argue that CAFE forwarding is scalable since the upper bounds of forwarding key length is in similar range of IPv6 header, which is considered scalable.

4.4 Conclusions

In this chapter, we have presented numerical results corresponding to the effectiveness of the content resolution and the content delivery processes. The results confirmed that there is no critical element in the COMET system, which could lead to the system performance degradation. More specifically, we can state the following:

- The execution delays of the content resolution process in highly loaded COMET entities are on the level satisfying the customers (the CRT delay is less that 2.5 s for 95 percentile of content requests);
- Even when the system size increases (content request rate, number of content sources, domain size), the CRT delays are still acceptable;
- Overhead of the COMET header, even in the worst case, is in similar range as the IPv6 header;
- The CAFE performances are similar to the performances of the IP router.

Concluding, from the point of view of handling the content resolution process and the content delivery process there are no barriers in deploying the COMET system in large-scale networks, like the Internet.

We can accelerate the execution of the content resolution procedure by using caches and enhanced algorithms for the retrieval of server loads and paths. The reported results indicate that:

- Caching mechanisms used in the client CME for storing content/authoritative records, paths and server load may essentially reduce CRT delay because the client CME avoids communication with other COMET entities;
- SQA (or similar) algorithm for the retrieval of server loads and paths may reduce the CRT delay especially for the case of large number of content sources.

Therefore, we recommend applying both of the discussed mechanisms in the COMET system.

From the point of view of the system scalability, it is important that in the decoupled approach, the control of the content resolution process is performed by the client CME and the server CME, both of them located at the network edge (not in the transit domains). Hence, if we have problems with handing content requests in a given domain, we can offload the COMET entities by adding new instances in this domain. This is a very positive feature of the discussed system supporting its scalability. So, one can find some similarities with DiffServ architecture, which is regarded as a reference scalable solution. In this architecture, the most important is that the main processes are performed by edge routers while the core routers perform only forwarding. When traffic grows, we simply add new edge routers without changing the core.

5 Scalability and performance of the COMET coupled approach

5.1 Introduction

Contrary to the decoupled approach, the coupled approach resolves content in a blind hop-by-hop manner, following certain rules when an AS does not contain knowledge about a requested content. Given the lack of centralised global knowledge of the network locations of contents, we are interested in determining the impact that this may have on the content resolution time and the characteristics of the content delivery paths, and whether paths become unnecessarily long as a result. Accordingly, this section presents simulation results carried out using real Internet topology data captured by CAIDA [15]. For the case when a CRME cannot resolve a content item, we propose two different rules which ASes can follow: a broadcast and a random mode. We explain each of these modes in the next section, and then present the performance evaluation results of the coupled approach in Section 5.3.

5.2 Content Resolution Modes

When a content client requests content from its local CRME, if the CRME cannot find a record for the requested content in its content table, it will forward the request to the CRME in its provider AS. In case there are multiple providers (i.e. the AS is multi-homed), the CRME may use one of two resolution techniques: a *broadcast* or *random* scheme.

In the broadcast scheme, the CRME sends the content request to the CRMEs in all provider ASes. Each provider CRME will also carry out the same policy, if it cannot find a record for the content in its content table. On the other hand, under the random scheme, the CRME will choose a provider CRME at random to which to forward the request. An example of content resolution under both schemes is shown in Figure 17. Here, a content client at AS6 requests a content that happens to be hosted at a content server in AS3.

In broadcast mode, AS6 will forward the request to both of its provider ASes (4 and 5). AS4 will, in turn, forward the request to both of its provider ASes (1 and 2), while AS5 will forward to its provider AS (2), and AS2 to its provider AS (0). Since the CRMEs in ASes 0 and 1 contain entries in their content table for the requested content, they will route the content request downhill towards the content server. Therefore, in this example, the broadcast scheme requires that a content request traverse seven ASes in total.

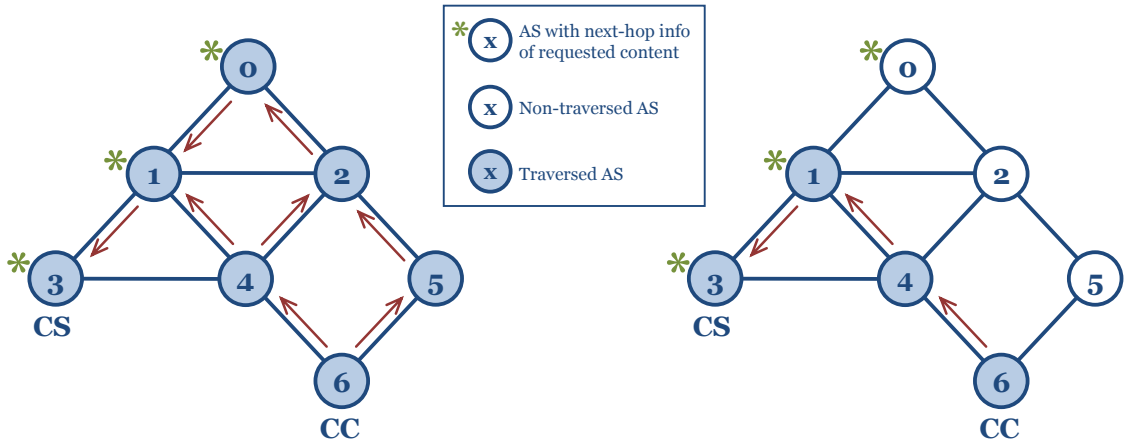


Figure 17: Example of the (a) broadcast, and (b) random resolution modes.

In random mode, AS6 randomly selects AS4 as the next uphill AS to which to forward the content request, which in turn will select AS1. AS1 will then forward the content request downhill to the content server according to the entry in the CRME. Therefore, in this example, the random scheme requires 4 ASes to be traversed.

5.3 Performance Evaluation

We analysed the performance of the CURLING system [12] using both Broadcast and Random resolution schemes outlined in the previous section. The aim of the evaluation was to determine how well CURLING performs in an Internet-based topology, and to determine if the hop-by-hop resolution approach will lead to suboptimal delivery paths.

5.3.1 Simulation Setup

We developed a custom Java simulator, and evaluated the performance of CURLING under various topology sizes. For each topology, we allocated 100 ASes to hold a single content server, where each server hosts a single unique content. We also randomly allocated 1000 content clients requesting content from the content servers based on a power-law distribution (Zipf with $\alpha = 1$).

The topologies used are single-rooted sub-topologies of a CAIDA-measured datasets [15] from the year 2010, with AS 701 (Verizon) as the root AS. We varied the number of ASes per topology from 400 to 1400 in increments of 200. The ratio of peering links to multi-homed ASes maintained in the sub-topologies was 1:3.61 – the same ratio present in the full 2010 CAIDA topology from which the sub-topologies were derived. We also aggregated lower tiers such that the sub-topology consists of three tiers, since lower tiers are known to lack certain relationship details amongst ASes [18].

5.3.2 Empirical Results

5.3.2.1 Content Resolution Overhead

The content resolution overhead is defined as the number of ASes that must be traversed in order to resolve a content from a server. Figure 18 shows the expected number of traversed ASes for different sizes of sub-topology. As can be seen, the random scheme requires between 3.9 and 4.4 AS traversals on average across all simulated sub-topology sizes, whereas the broadcast scheme between 4.7 and 5.2 AS traversals. Relative to the random scheme, the broadcast scheme leads to a considerable increase in the number of ASes traversed to resolve a content. Across all simulated topologies, the average increase is 20% with a variance of 1.6%. This highlights the fact that AS traversals have no dependence on the topology size, therefore, the percentage increase in AS traversals is likely to be the same in a large Internet-scale topology.

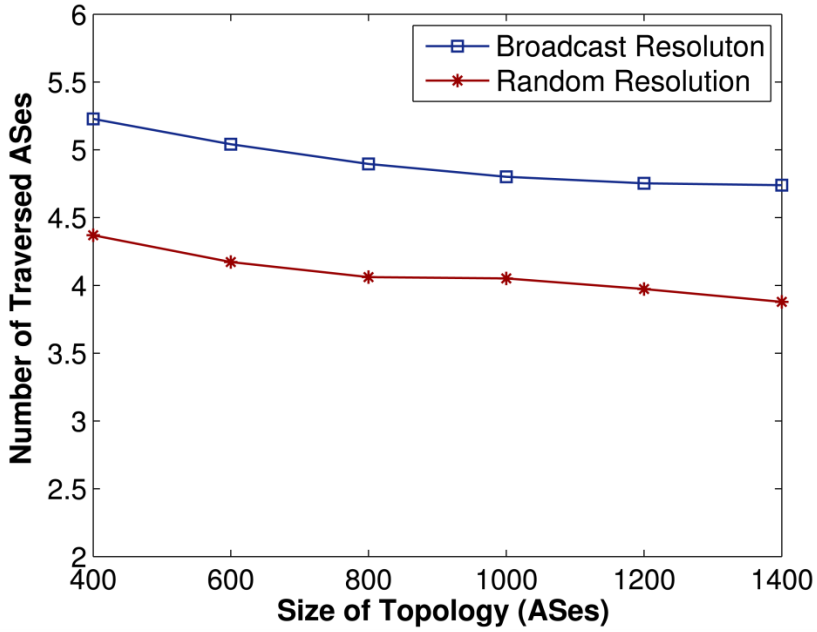


Figure 18: Number of traversed ASes during content resolution for random and broadcast resolution schemes.

5.3.2.2 Content Resolution Path Length

Given the increased AS traversals of the broadcast scheme over the random scheme, we are interested to see whether there are any benefits the broadcast scheme can offer in return for its added overhead, in terms of the content delivery path length (hop count). Figure 19 shows the expected hop count for different sizes of sub-topology. The *resolution* path plots shown are essentially the delivery paths before BGP route optimisation, since these are simply the reverse of the paths taken during the resolution process. On the other hand, the *delivery* path plots relate to the paths *after* BGP route-optimisation.

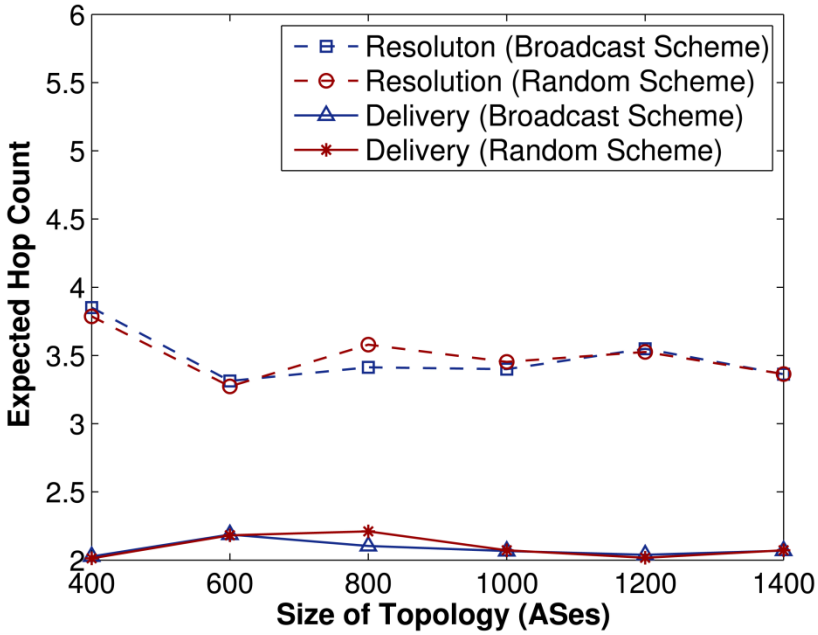


Figure 19: Expected hop count of non-optimised resolution path and route-optimised delivery path for random and broadcast resolution schemes.

As can be seen from the Figure 19, the hop count prior to BGP route optimisation, the delivery path length varies from 3.3 to 3.9 hops, whereas after optimisation, the hop count varies from 2.0 to 2.2 hops. This represents a 40% reduction in the number of hops after optimisation. Similar results, leading to an identical 40% reduction in the hop count is found for the case of the random scheme. However, more importantly, comparing the broadcast scheme with the random scheme it can be seen that the path length of the random scheme is virtually the same as the broadcast scheme, both before and after route-optimisation and across all simulated topologies. This leads to the conclusion that the broadcast scheme offers no tangible benefits over the random scheme to justify the extra overhead that it generates during content resolution. Such a conclusion is corroborated by the plots of hop count distribution for a 1000-size sub-topology shown in Figure 20, in which it can be seen that the distribution of both non-optimised path lengths [Figure 20(a)] and optimised paths lengths [Figure 20(b)] are both respectively very similar.

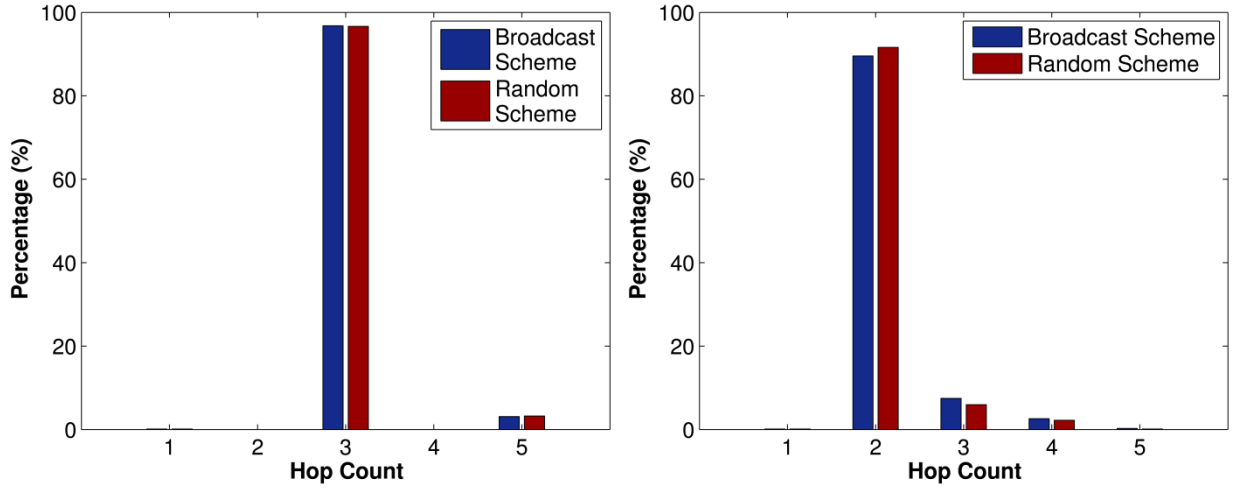


Figure 20: Distribution of hop count in a 1000-size CAIDA sub-topology for random and broadcast resolution schemes along (a) the non-optimised resolution path, and (b) the route-optimised delivery path.

5.3.2.3 Content Access Time

The content retrieval time is defined as the duration of time from the moment a content client issues a request for a particular content to the moment it starts to receive it. It is influenced by three main factors: the inter-AS propagation time, the intra-AS propagation time, and the CRME content table lookup time. Since the maximum resolution hop count, $N_{interAS}$, was found to be 5 hops, we can use this value to determine a worst-case content access time, using approximations of propagation time published in the literature [40],[41]. Specifically, we use the latency approximations, $D_{interAS}$ and $D_{intraAS}$ of 34ms for inter-domain hops and 2ms for inter-domain hops, respectively [40], and the number of intra-domain hops to be $1 + \lfloor \log D \rfloor$ where D is the degree of the domain [41]. Based on this we can formulate the worst-case content retrieval time to be:

$$T_{CR} = 2 \cdot N_{interAS} \cdot D_{interAS} + (N_{interAS} + 1) \cdot (2 \cdot D_{intraAS} \cdot [1 + \lfloor \log D \rfloor] + T_{LU}) \quad (2)$$

where T_{LU} is the content table lookup time, which we assume to be no more than 5ms , see assumptions in Table 2.

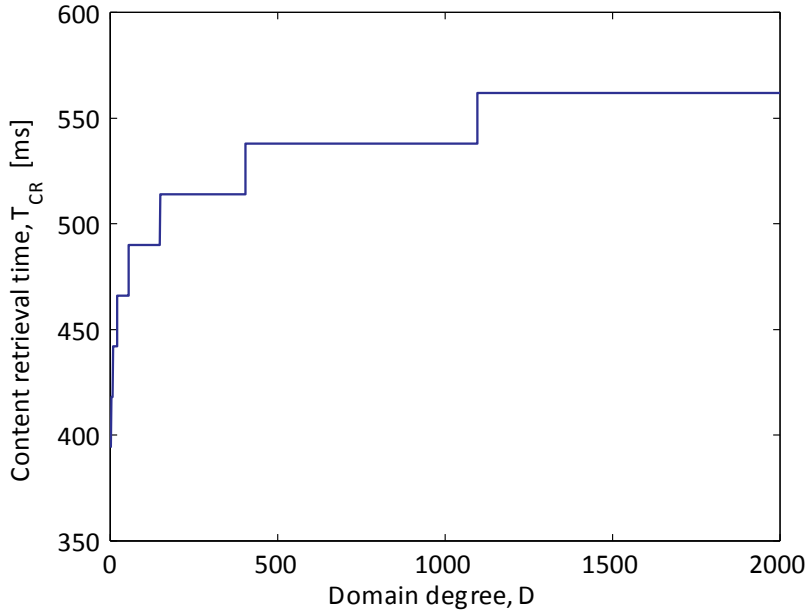


Figure 21: Content retrieval time as a function of domain degree.

We vary *D* and plot the resultant content access time in Figure 21. From this figure, it can be observed that if each AS were to have a degree of 2000, the content retrieval time 562ms, which is significantly smaller than maximum tolerable content retrieval time of 3s specified in [7]. In reality, domains with degrees of connectivity as high as 2000 are typically limited to tier-1 ASes; lower-tier ASes will typically have much lower degrees, therefore the content retrieval time is expected to be a lot less than. Part of the reason for the low content retrieval time in comparison from the maximum tolerable time comes from the way in which the coupled approach carries out content resolution in a single round trip.

5.4 Conclusions

Based on the performance evaluation results of the coupled approach, we can conclude that the coupled approach can resolve content requests in a scalable manner. The final resolution path under both Random and Broadcast resolution modes is virtually the same on average, meaning that the higher number of AS traversals under Broadcast mode compared to Random mode leads to no justification for the use of Broadcast mode. This argument is further reinforced by the fact that the average hop count of both resolution modes *after* route optimisation is also virtually the same. It was also shown that the average hop count along the resolution and optimised delivery paths does not seem to vary with topology size. Therefore, looking at the deployment of the coupled approach at an Internet-scale (purely from a scalability point-of-view), the results suggest that using even a random resolution scheme, the delivery path length is kept to within five hops. Consequently, the content retrieval time can be kept to well under one second, in part because of the single round trip required by the coupled approach to resolve content.

6 In-network caching

6.1 Introduction

Naming content objects directly and routing to those objects, instead of the machines that host the requested objects, gives the opportunity to identify contents as they travel from source to destination [42]. In turn, given that the network transfers *named* objects (instead of unidentifiable *data containers, i.e.*, IP packets), these objects can be cached *in the network* and be forwarded to subsequent users interested in the same content.

In-network caching has therefore emerged as a distinct research field in the context of *Information-Centric Networks*. ICN enables caching of addressable content *chunks* in *every* cache-equipped network device and replacement of cached chunks at line-speed. Although, this operation increases the availability of cache locations it renders prohibitive the process of updating logically-centralised content location databases with the exact location of cached contents. This decentralised, location-independent operation alters many of the basic features of past overlay caching techniques, *e.g.*, content-to-cache allocation, while it invalidates the applicability of some others, *e.g.*, content placement based on fixed overlay topologies of caches and servers.

In the next two subsections, we present the performance evaluation results of the two main in-network caching strategies developed in COMET. Namely, the Probabilistic In-Network Caching algorithm initially presented D4.2 [4] and in [43] and the Centrality-based approach to caching presented also in D4.2 [4] and in [43]. In the context of the COMET system, the proposed algorithms run at the CMP level and in particular, it is the Content Mediation Function (CMF) which makes decisions as to which contents should be cached along the delivery path. Then, the CMF instructs the newly introduced Cache-Aware Caching Function (CACF), which in turn acts accordingly and caches incoming contents. CACF is introduced in D2.3 [2] as a new function of the CFP. The detailed description of how content objects are split into content chunks and are distributed along the network caches is given in D2.3 [2]. We note that in the COMET architecture, caching takes place only in CAFEs and not in every network router of a domain.

6.2 Probabilistic In-network Caching

In this study, we are concerned with cache management operations that have to be adjusted to fit in a completely decentralised and uncoordinated environment. We focus on the allocation of the available cache capacity along a *path* of caching entities among different content flows. As a starting point, we intuitively observe that caching *every* content in *every* cache-enabled device along the delivery route (caching behaviour proposed in [42]), inherently causes huge *caching redundancy*. Subsequently, our goal is to reduce caching redundancy and make more efficient use of available cache resources, in order to reduce overall network utilisation and increase user-perceived quality. Our initial investigation of selective caching policies based on node centrality metrics yielded very promising results in this direction [4]. Details regarding this study on centrality-based caching is presented in section 6.3.

To achieve our goal, we approximate the *caching capability* of a given path *per unit time* and we design *ProbCache*, a *probabilistic algorithm* for distributed content caching and fair content multiplexing along a path of caches (D4.2 [4] and [43]). In particular, *ProbCache* allocates caching space to content flows based on the number of hops from source to destination. That is, flows connected close to the content source will be given priority to cache in the available caches along their short route, over content flows that travel further away and which have the opportunity to cache their contents in other nodes along the delivery path. Content multiplexing fairness is therefore, associated with the amount of caching resources along a path of caches that a specific content flow utilises.

ProbCache was initially introduced in [43] and in D4.2 [4]. In the present study, we elaborate on the content flow multiplexing fairness of *ProbCache* and find that although it

significantly improves the overall network performance (*e.g.*, in terms of cache hits), it sometimes fails to multiplex content flows in a fair manner. We apply modifications to the original design of *ProbCache* in Section 6.2.1 and then both analytically and through simulations, we evaluate the performance of the enhanced version of the algorithm.

We evaluate the performance of both versions of *ProbCache* under several simulation setups. We compare the proposed algorithm against well-known caching approaches, such as universal caching and Leave Copy Down (LCD) [44]. Our results suggest that there is indeed a lot of space for resource management optimisation of in-network caching policies, given that appropriate resource allocation rules are in place (see our results later in this section). Our results show that careful content flow multiplexing in caches can achieve up to 20% more cache hits in case of small scale flash crowd events and an average of 11-13% under normal conditions. Surprisingly, this translates to an order of magnitude reduction in terms of traffic redundancy. We define a *Content Multiplexing Fairness Index* to better capture the resource allocation properties of the protocols and show that *ProbCache* exhibits desirable properties in that respect as well.

We use the terms “router", “cache" and “node" interchangeably to refer to cache-enabled network devices; it should be noted that our approach does not require every router to be cache-enabled, but it will work in hybrid architectures as well. Furthermore, we refer to content “messages" and “chunks" interchangeably to refer to the *cacheable unit*, which is not necessarily of similar size to an IP packet. In fact, we leave open the actual size of the cacheable unit which is yet to be defined by the ICN research community. We highlight that the concepts and algorithms proposed in this paper are *cache unit-* as well as *architecture-agnostic* and would apply to almost any ICN environment.

As mentioned the basic functionality of ProbCache was introduced in [43] and in [4]. We refer the reader to these studies for the description of the basic algorithm. The paper which came out of our investigations given below is also attached together with this deliverable as Annex A. We therefore, take as granted the knowledge of the basic algorithm and discuss below the extensions for Heterogeneous cache-size environments. We then provide a quick summary of our preliminary evaluations presented in [43] and finally, present our extensive analysis and results.

6.2.1 Extending *ProbCache* for Heterogeneous Cache Sizes

We expect that as research in the area matures, caches will be sized according to a specific norm or a set of guidelines. We conjecture three potential directions: *i)* larger caches are deployed towards the core of the network, where servers reside, *ii)* larger caches are deployed towards the edges of the network, where users are connected, *iii)* all caches have roughly similar sizes.

For ease of illustration, we assume that caches are sized according to the formula: $N_i \times TSB$ (see Table 2 in [43]); that is, we set the *unit cache*, N_i, to be one second worth of traffic and assign to each router the amount of cache that corresponds to its distance from the closest source[1] in terms of hops (*i.e.*, TSB). We modify the *TimesIn* and *ProbCache* functions accordingly to comply with different cache-size settings along the path. For example, if a core router caches one second worth of traffic and we assume larger caches towards the edges of the network (which we denote as $c(ore) \rightarrow E(dge)$), then the edge router of a six-hop path will cache six seconds worth of traffic. The capacity of the path in that case is $c \rightarrow E: \sum_{i=1}^{c-(x-1)} (c-i)N_i$. The opposite applies for deployments where larger caches are deployed in the core of the network (which we denote as $C(ore) \rightarrow e(dge)$); the capacity of the path in this case is $C \rightarrow e: \sum_{i=1}^{c-(x-1)} iN_i$. Finally, when roughly equal or random size caches are deployed all throughout the network, the capacity of the path is $c \rightarrow e: \sum_{i=1}^{c-(x-1)} N_i$.

[1] The source is here largely defined as the area were servers are connected (core or backbone network). Although this might not be realistic as a setting in real deployments, since a router is connected to multiple servers at different distances, we consider that similarly to different-tier domains, router caches can be sized according to the tier they belong too.

Based on the path capacity in each case, we modify the $TimesIn(x)$ factor and the $ProbCache(x)$ formula (denoted as $P(x)$). The corresponding Equations are shown below.

Equal-sized caches along the path ($c \rightarrow e$):

$$TimesIn_{c\rightarrow e}(x) = \frac{\sum_{i=1}^{c-(x-1)} N_i}{T_{tw}N_x} = \frac{N_i(c-x+1)}{T_{tw}N_x} \text{and} P_{c\rightarrow e}(x) = \frac{N_i(c-x+1)}{T_{tw}N_x} \times \frac{x}{c} \quad (3)$$

Larger caches in the core ($C \rightarrow e$):

$$TimesIn_{C\rightarrow e}(x) = \frac{\sum_{i=1}^{c-(x-1)} iN_i}{T_{tw}N_x} = \frac{N_i x(c-2x+2)}{T_{tw}N_x} \text{and} P_{C\rightarrow e}(x) = \frac{N_i x(c-2x+2)}{T_{tw}N_x} \times \frac{x}{c} \quad (4)$$

Larger caches towards the edge ($c \rightarrow E$):

$$TimesIn_{c\rightarrow E}(x) = \frac{\sum_{i=1}^{c-(x-1)} (c-i)N_i}{T_{tw}N_x} = \frac{1}{2}\frac{N_i}{T_{tw}N_x}(c^2 - c - x^2 + 3x - 2) \quad (5)$$

$$P_{c\rightarrow E}(x) = \frac{1}{2}\frac{N_i}{T_{tw}N_x}(c^2 - c - x^2 + 3x - 2) \times \frac{x}{c} \quad (6)$$

In Figure 22, we plot the *path cache capacity* as it is calculated by *each node* along a six-hop path for Eqs. 3, 4 and 5.

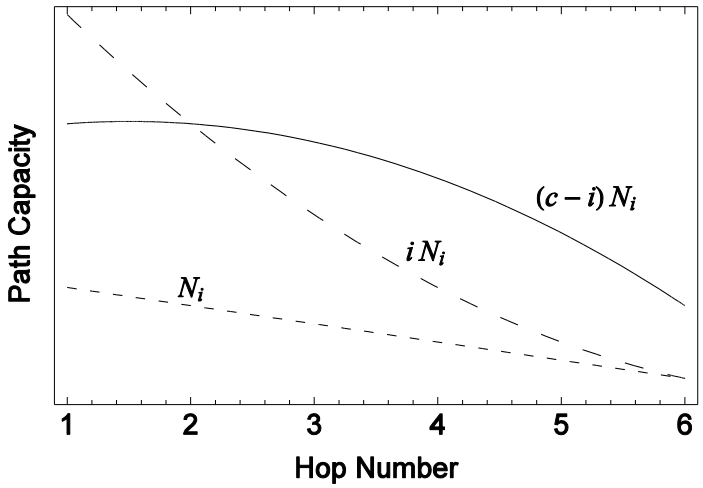


Figure 22: Path Capacity calculation as performed by each router along a 6-hop Path: the N_i curve is for homogeneous caches, hence the capacity is given by $c \rightarrow e$: $\sum_{i=1}^{c-(x-1)} N_i$; the iN_i curve is for larger core caches, hence $C \rightarrow e$: $\sum_{i=1}^{c-(x-1)} iN_i$; the $(c-i)N_i$ curve is for larger caches towards the edge, hence $c \rightarrow E$: $\sum_{i=1}^{c-(x-1)} (c-i)N_i$.

6.2.2 Summary of (Preliminary) Evaluations

In [43] and in D4.2 [4], we have evaluated the performance of *ProbCache* (Eqs. 3, 4, 6) in the above three different settings (*i.e.*, equal caches along the path, larger in the core and larger towards the edge). We have used binary-tree topologies and compared the performance of *ProbCache* against alternative algorithms and caching approaches proposed in the related literature (*e.g.*, universal caching used in [42] and LCD [44]).

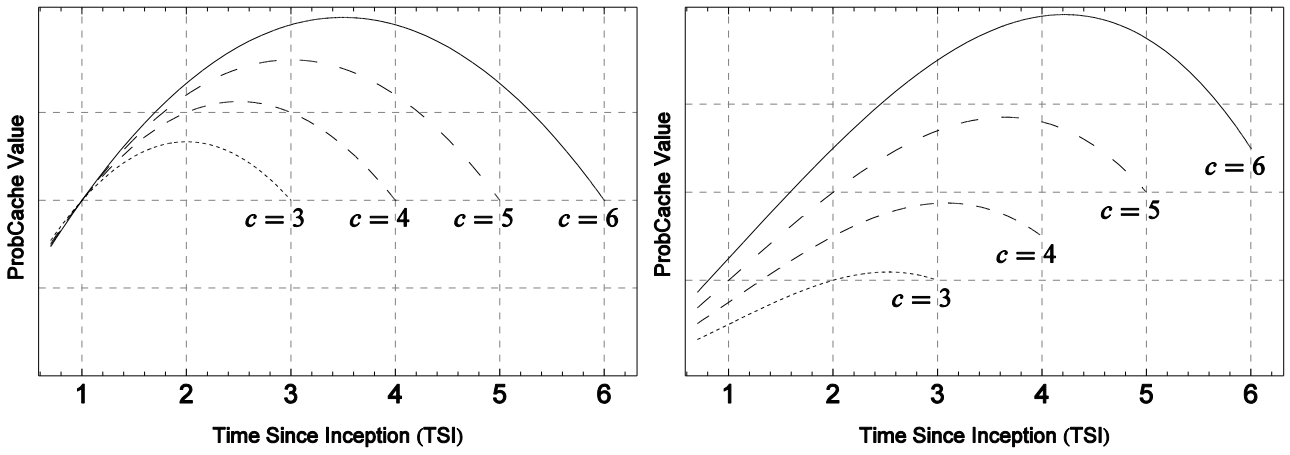
In case of homogeneous caches along the path, our results showed a reduction of up to 20% in server hits, and up to 10% in the number of hops taken to hit cached contents, but, most impressively, reduction of cache evictions by an order of magnitude, compared to universal caching. In case of heterogeneous caches along the delivery path, we found that when we deploy more cache capacity in the core of the network the algorithms do not seem to realise and exploit the extra resources available. In contrast, when larger caches are put towards the edge of the network and closer to the end-users, *ProbCache* exploits most of the extra available resources along the delivery path and improves significantly the overall network performance as perceived by the end-user. We refer the reader to [43] for a more detailed discussion of the results, which we omit here due to space limitations.

Based on the results reported in [43] and briefly summarised above, for the rest of this paper we elaborate on the performance of *ProbCache* under two cache-size settings, that is, equal-sized caches along the whole path (Eq. 3) and larger caches towards the edge (Eq. 6).

6.2.3 Analysis I

We elaborate on the performance of *ProbCache* and in particular on the version of the algorithm presented in Eqs. 3 and 6. We refer to those two versions of the formula as $P_{c\to e}(x)$ and $P_{c\to E}(x)$, where x is the TSB value, *i.e.*, the current distance of the content chunk from the source in terms of hops. The intention is to explore the behaviour of the algorithms with regard to their resource management and utilisation properties. We are interested in *fair and efficient* content flow multiplexing in in-network caching architectures, which would in turn increase the overall network performance.

We begin by plotting the behaviour of the two versions of *ProbCache* for users connected at different points along an example six-hop path.



[Behaviour of $P_{c\to e}(x)$, Eq. 3.] [Behaviour of $P_{c\to E}(x)$, Eq. 6.]

Figure 23: Behaviour of $P_{c\to e}(x)$ and $P_{c\to E}(x)$ along a six-hop long delivery path. We observe that the longer a flow's distance is from the content source, the higher the chances it has got to cache its contents along the whole path. This is in contrast to our design considerations for fair content multiplexing.

We observe that although fair content multiplexing was one of the main design considerations for $ProbCache$, and the main operational property of $CacheWeight$, in reality, the path capacity calculation ($TimesIn$ factor) has unexpectedly high impact on the overall behaviour of the algorithm. That is, we see that the longer the distance from a client to the source of content, the higher the probability this client has to cache contents along *the entire path* (*i.e.*, even far from its attachment point and closer to the source; nodes 1 to 4 in Figure 22). Clearly, this behaviour leads to unfairness in terms of resource allocation between clients. To verify this observation get a deeper understanding of the operational properties of the algorithm, we elaborate on the distribution of values that the function gets along a path of caches for clients connected at different points along the path. We initially calculate the derivatives of the two versions of the algorithm to see at which point along the path $ProbCache$ gets its maximum values; we also calculate the integrals of $ProbCache$ to monitor the density distribution of the value range of $ProbCache$ along the delivery path.

6.2.4 Maximum Values of Basic Functions

In this section, our intention is to find the number of hops away from the content source that $ProbCache$ gets its maximum value for users connected at different points along the path.

6.2.4.1 $P_{c\to e}(x)$

We begin with the basic case, where we assume equal size caches along the path (Eq. 3):

$$P_{c\to e}(x) \quad = K_1(c - x + 1)\frac{x}{c}$$

where $K_1 = \frac{N_i}{T_{tw}N_x}$, which we omit in the equations from now on, since being a constant does not affect the overall result.

We set the derivative of the basic function to zero to find the points where the function is maximised:

$$\frac{dP_{c\to e}(x)}{dx} = 0 \Rightarrow \frac{d}{dx}\left((c - x + 1)\frac{x}{c}\right) = 0 \Rightarrow \frac{c-2x+1}{c} = 0 \Rightarrow x = \frac{1+c}{2} \qquad (7)$$

6.2.4.2 $P_{c\to E}(x)$

We repeat the same process for the version of $ProbCache$ that applies in case of increasing cache sizes towards the edges of the network (Eq. 6):

$$P_{c\to E}(x) \quad = K_2(c^2 - c - x^2 + 3x - 2)\frac{x}{c}$$

where $K_2 = \frac{1}{2}\frac{N_i}{T_{tw}N_x}$, which we omit in the rest of the analysis, as it does not affect the final result.

The derivative of $P_{c\to E}(x)$ is given below.

$$\frac{dP_{c\to E}(x)}{dx} = \frac{d}{dx}\left((c^2 - c - x^2 + 3x - 2)\frac{x}{c}\right) = \frac{c^2-c-3x^2+6x-2}{2c} \qquad (8)$$

We equal the derivative of $P_{c\to E}(x)$ to zero, in order to find the *local maxima* of $ProbCache$.

$$\frac{dP_{c\to E}(x)}{dx} = 0 \Rightarrow |x - 1| = \frac{\sqrt{c^2-c+1}}{\sqrt{3}} \qquad (9)$$

which finally gives us the following:

$$x = \frac{1}{3}\left(3 - \sqrt{3}\sqrt{1 - c + c^2}\right) \quad \text{or} x = \frac{1}{3}\left(3 + \sqrt{3}\sqrt{1 - c + c^2}\right) \tag{10}$$

Out of these two roots of the equation, the second one is the one with positive values, hence, the one we are interested in.

6.2.4.3 Observations

We plot the derivative of $P_{c\to e}(x)$ (Eq. 7) and of $P_{c\to E}(x)$ (Eq. 10) in Figure 24 for users connected at several different points along the sample 6-hop delivery path. The x-axis denotes the distance of the client from the source node (*i.e.*, the TSI value - see Table 2 in [43]), while the y-axis denotes the hop number (*i.e.*, TSB) for which $ProbCache$ gets its highest value for each different value of TSI.

We observe that $P_{c\to e}(x)$ (function 1 in Figure 24) gets its maximum values half-way through the path from the source to the client, while $P_{c\to E}(x)$ (function 2 in Figure 24) is maximised just after the middle of the delivery path. This was also shown in Figure 22 and Figure 23. The grey area in Figure 24 denotes the area where $ProbCache$ should be maximised, in order to guarantee fair resource allocation between clients. In particular, and as mentioned before (e.g., in D4.2 [4]), we argue that caching space should be allocated to users in proportion to their distance from the source. Proportional resource allocation translates to maximisation of $ProbCache$ within the grey area in Figure 24, that is, close to the flow's destination. In this figure, we observe that when clients are connected one, two or three hops away from the source the value of $ProbCache$ is maximised within the desired grey area, while for clients connected further away, the maximum value of the function falls outside this area.

However, as shown in Figure 22, even for small values of TSI, where $ProbCache$ takes its maximum value within the grey area, we see that this maximum value is still smaller for these flows than for instances of $ProbCache$ where TSI gets higher values (*i.e.*, for users connected further away from the source node). In turn, this gives constant advantage in terms of caching probabilities to flows connected more hops away from the core of the network. Therefore, we argue that the maximum value of the caching function can not be a measure on its own, but instead it has to be evaluated together with the *density distribution*[2] of the function for each particular instance. The density of the function with regard to the TSI and TSB values of $ProbCache$ is given by its integral, which we calculate next in order to verify our claims.

[2] The density of the function here is similar to the *expectation* of a probability distribution function of a random variable. However, given that $ProbCache$ may take values higher than 1, which would mean that the path cache resources allow for caching of multiple copies of the chunk in question, $ProbCache$ is not a probability distribution function (and x not a random variable). Therefore, to find the *density distribution* of $ProbCache$, we calculate its integral.

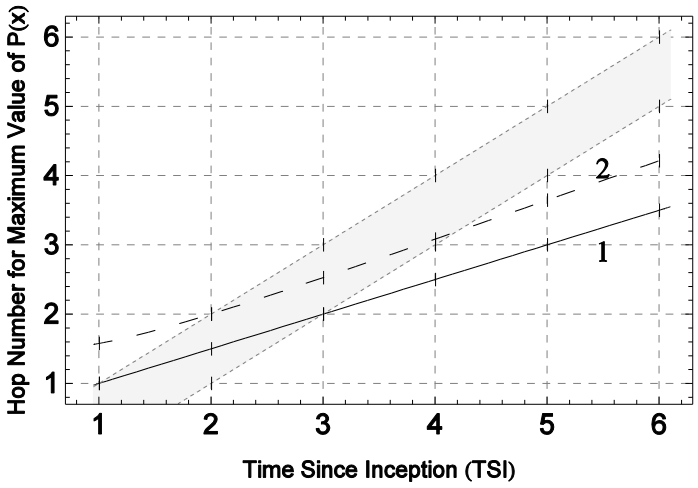


Figure 24: Derivatives of: (1) $P_{c \to e}(x)$ (Eq. 7) and (2) $P_{c \to E}$ (Eq. 10). Grey area shows the ideal interval where maximum values for $ProbCache$ would guarantee fair resource allocation along the delivery path.

6.2.5 Density Distribution of Basic Functions

Integrals capture the density distribution of $ProbCache$ with regard to the values that the function takes for different distances of clients from the source. The distribution of the density relates closely to the cache deployment setting along the path, that is, the caching resources available along the path. For example, in the heterogeneous cache size scenario, where more cache is deployed towards the edges of the network, we expect that the distribution density of $ProbCache$ will increase as we move away from the source of content (*i.e.*, for clients connected far away from the source). This is because these clients have more resources available to cache and therefore, $ProbCache$ is more likely to choose these nodes to cache contents of clients travelling longer paths, in order to leave nodes in the core for other content flows.

In contrast, in case of equal caches along the path, we expect that all users have equal opportunities to utilise the available resources irrespective of their distance from the source. Hence, in this case the distribution density should be somewhat evenly distributed along the path.

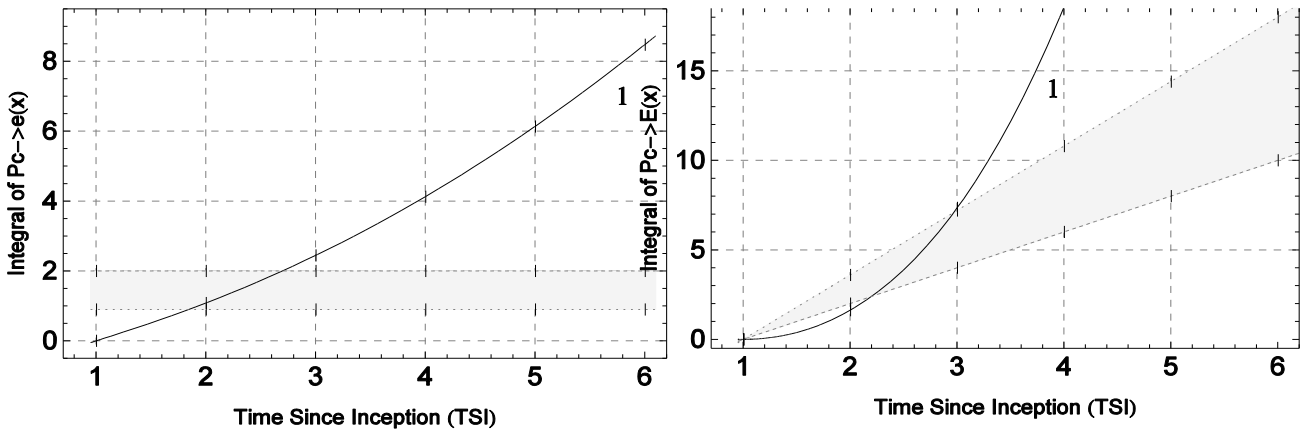
To depict this situation and verify our claims, we initially calculate and then plot the integrals of $P_{c \to e}$ (Eq. 3) and of $P_{c \to E}$ (Eq. 6).

6.2.5.1 Integral of $P_{c \to e}(x)$

$$\int_1^c P_{c \to e}(x) = \int_1^c \frac{(1+c-x)x}{c}\, dx = \frac{c^2}{6} + \frac{c}{2} - \frac{1}{6c} - \frac{1}{2} \tag{11}$$

6.2.5.2 Integral of $P_{c \to E}(x)$

$$\int_1^c P_{c \to E}(x) = \int_1^c \frac{x\left(-2-c+c^2+3x-x^2\right)}{c}\, dx = \frac{1}{2} + \frac{1}{4c} - \frac{3c}{2} + \frac{c^2}{2} + \frac{c^3}{4} \tag{12}$$



[Density distribution of $P_{c\to e}(x)$, Eq. 3. The integral of this function given in Eq. 11.]

[Density distribution of $P_{c\to E}(x)$, Eq. 6. The integral of this function given in Eq. 12.]

Figure 25: Density distribution, in terms of the integrals of the two versions of *ProbCache* for homogeneous and heterogeneous cache deployment scenaria. The grey area covers the ideal area, within which the algorithm would guarantee fair cache allocation among content flows.

6.2.5.3 Observations

We plot the integrals of the two functions as calculated above (Eqs. 11 and 12) in Figure 24. In the same Figure, we also sketch the available *cache capacity distribution* along the path - the grey area - in each of the two cases. The cache capacity distribution is calculated taking into account the fact that clients should cache their contents close to their point of attachment (*e.g.*, in the last two hops of the delivery path) with higher probability (as also shown in Figure 24). For the homogeneous cache deployment case, for instance, the capacity distribution is the same along the entire path, denoting that all users should have equal chances to cache regardless of their position on the path. Instead, in the heterogeneous cache deployment case, the capacity distribution is increasing as it takes into account the extra cache resources available towards the edge of the network. We contend that the distribution of the value of *ProbCache* should be within this grey area in order to achieve fair content multiplexing in in-network caching environments.

We observe, however, that *ProbCache* does not achieve distribution density close to the "fair" grey area in any of the settings evaluated. Instead, in both cases, $P_{c\to e}(x)$ and $P_{c\to E}(x)$ appear more aggressive and with density distributions that favour users connected far from the source. This fact verifies our earlier claim, which we presented in Figure 22.

Based on our observations, we proceed to deal with this unfair behaviour by modifying the $CacheWeight$ factor of *ProbCache*. We introduce our modification and assess its performance in the next section.

6.2.6 Enhancing the content multiplexing fairness of *ProbCache*

To improve the content multiplexing fairness of *ProbCache* (Eqs. 3 and 6), we raise the value of $CacheWeight$, $\frac{x}{c}$), to the power of c (*i.e.*, the TSI value of the delivery path), $(\frac{x}{c})^c$. For this modification, we reason as follows. Figure 22 shows that *ProbCache* increases the probability of contents being cached very early compared to the distance of the client to the source. In contrast, we argue that the ideal algorithm would increase its value directly proportionally to the distance of the client from the source. That is, it would get low values when going through the first nodes of the path and would increase as it would get closer to its destination.

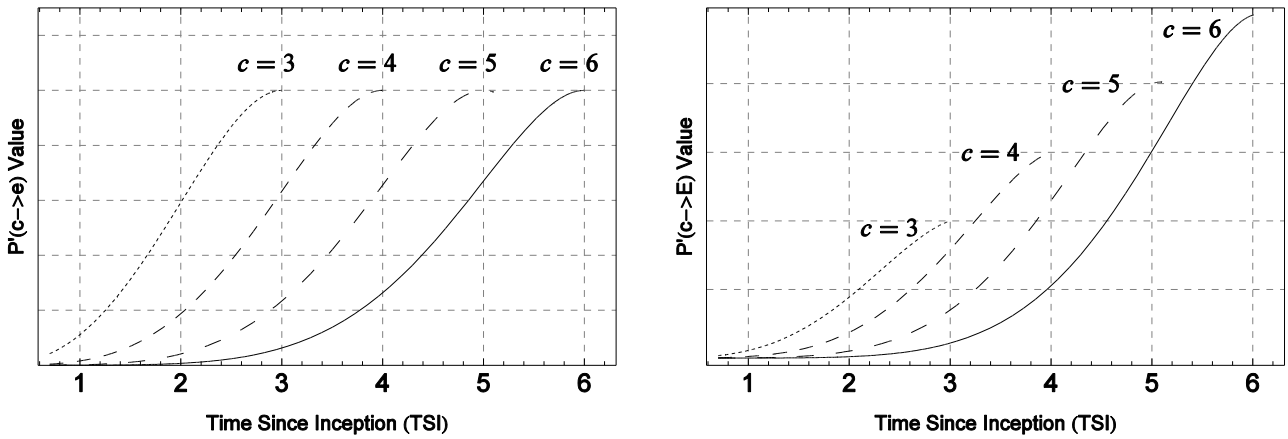
Our modified *ProbCache* functions, which we denote as $P'_{c\to e}(x)$ and $P'_{c\to E}(x)$ for the homogeneous and the heterogeneous case, respectively are therefore, the following:

$$ProbCache'_{c\to e}(x) = \frac{N_i(c-x+1)}{T_{tw}N_x} \times (\frac{x}{c})^c \Rightarrow P'_{c\to e}(x) = K_1(c-x+1)(\frac{x}{c})^c \quad (13)$$

where $K_1 = \frac{N_i}{T_{tw}N_x}$, and

$$ProbCache'_{c\to E}(x) = \frac{1}{2}\frac{N_i}{T_{tw}N_x}(c^2-c-x^2+3x-2) \times (\frac{x}{c})^c \Rightarrow P'_{c\to E}(x) = K_2(c^2-c-x^2+3x-2)(\frac{x}{c})^c \quad (14)$$

where $K_2 = \frac{1}{2}\frac{N_i}{T_{tw}N_x}$. As before, we omit K_1 and K_2 in the following.



[Behaviour of $P'_{c\to e}(x)$, Eq. 13] [Behaviour of $P'_{c\to E}(x)$, Eq. 14]

Figure 26: Behaviour of $P'_{c\to e}(x)$ and $P'_{c\to E}(x)$ along a six-hop long delivery path. After the modification of the $CacheWeight$ factor (Eqs. 13 and 14), we now observe that the caching probability of clients increases directly proportionally to its distance from the source. This is in accordance to our initial design principles.

We plot the behaviour of these enhanced versions of $ProbCache$ in Figure 25. Clearly, the performance of the algorithms is different now and much closer to what we expected. Depending on the attachment point of the clients, the value of the algorithm adjusts accordingly, in order to leave resources closer to the source of the content for clients travelling shorter paths. In the following, we repeat the methodology followed before to verify our initial observations. In particular, we initially calculate the derivatives of both versions of the enhanced algorithm; we equal the derivatives to zero to find the maximum values of $P'_{c\to e}(x)$ and $P'_{c\to E}(x)$, which we plot in Figure 27; finally, we calculate and plot the integrals to evaluate the density distribution of the enhanced version of $ProbCache$.

6.2.7 Maximum Values of Enhanced Function

6.2.7.1 $ProbCache'_{c\to e}(x)$

The derivative of the enhanced function for the case of equal caches along the delivery path is given below:

$$\frac{dP'_{c\to e}(x)}{dx} = \frac{d}{dx}\left((c+1-x)\left(\frac{x}{c}\right)^c\right) = \frac{(c+1)(c-x)\left(\frac{x}{c}\right)^c}{x} = (c-x+1)\left(\frac{x}{c}\right)^{c-1} - \left(\frac{x}{c}\right)^c \quad (15)$$

We minimise the derivative of $P'_{c\to e}(x)$, in order to find its *local maxima*:

$$\frac{dP'_{c\to e}(x)}{dx} = 0 \Rightarrow (c - x + 1)\left(\frac{x}{c}\right)^{c-1} - \left(\frac{x}{c}\right)^{c} = 0 \Rightarrow x = c \quad (16)$$

6.2.7.2 $ProbCache'_{c\to E}(x)$

We follow the same route for $P'_{c\to E}(x)$.

$$\frac{dP'_{c\to E}(x)}{x} = \frac{d}{dx}\left((c^2 - c - x^2 + 3x - 2)\left(\frac{x}{c}\right)^{c}\right) = \frac{\left(\frac{x}{c}\right)^{c}(c^3 - c^2 - c(x^2 - 3x + 2) + (3 - 2x)x)}{x} \quad (17)$$

$P'_{c\to E}(x)$ is maximised according to Eq. 19:

$$\frac{dP'_{c\to E}(x)}{dx} = 0 \Rightarrow x = \frac{3(1+c)}{2(2+c)} - \frac{\sqrt{9+2c-7c^2+4c^3+4c^4}}{2|2+c|} \quad (18)$$

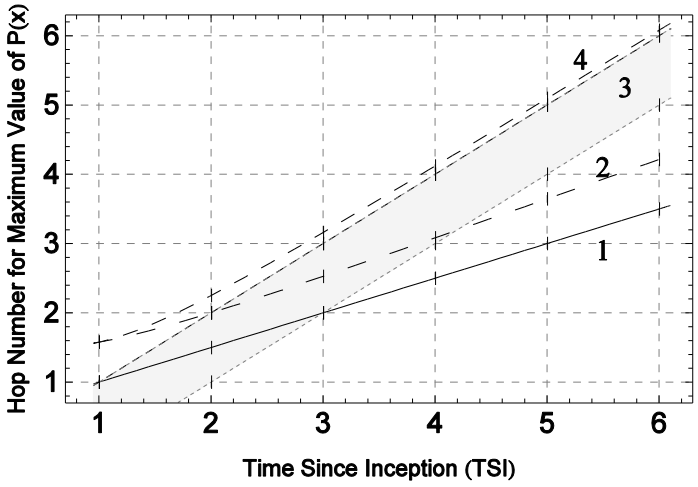


Figure 27: Derivatives of: (1) $P_{c\to e}(x)$ (Eq. 7), (2) $P_{c\to E}(x)$ (Eq. 10), (3) $P'_{c\to e}(x)$ (Eq. 17), (4) $P'_{c\to E}(x)$ (Eq. 18). In contrast to the original algorithm, the enhanced version gets its maximum values within the desired area, where fair content multiplexing is guaranteed, according to our design principles. This is the case for line-plots (3) and (4) in the above figure.

6.2.7.3 Observations

We plot the maximum values of $P'_{c\to e}(x)$ and of $P'_{c\to E}(x)$, according to Eqs. 16 and 18 in Figure 27. We observe that the enhanced versions of *ProbCache* for both the homogeneous and the heterogeneous cache deployments are behaving in a fairer manner. That is, the functions get their maximum values when contents go through nodes that are close to the content's destination, as can be seen by line-plots (3) and (4) in Figure 27 for $P'_{c\to e}(x)$ and $P'_{c\to E}(x)$, respectively. Clearly, this is a fairer behaviour than the one adopted by line-plots (1) and (2) in the above figure, which depict the performance of the original algorithm, as we also show through extensive simulations in the next Section.

6.2.8 Density Distribution of Enhanced Functions

As a final step, we calculate and plot the integrals of the enhanced versions of the algorithm. Our intention, as before, is to monitor the density distribution of the enhanced algorithm and see whether it falls within the grey area presented in Figure 24, which we consider as the ideal behaviour in terms of multiplexing between competing content flows along a path of caches.

6.2.8.1 Integral of $P'_{c\to e}(x)$

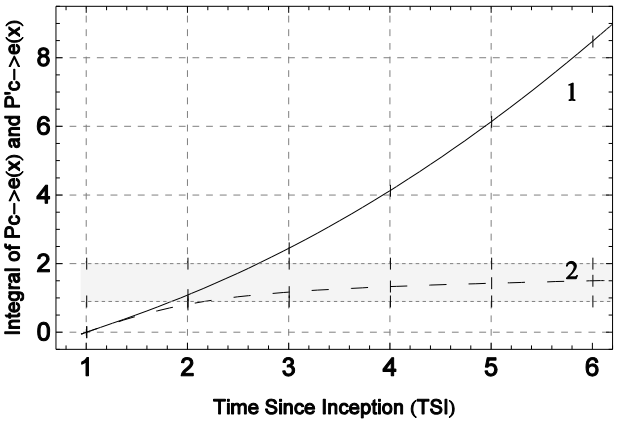
$$\int_1^c P'_{c\to e}(x) = \int_1^c (1+c-x)\left(\frac{x}{c}\right)^c \, dx = \frac{2c-(\frac{1}{c})^c(1+c)}{2+c} \tag{19}$$

6.2.8.2 Integral of $P'_{c\to E}(x)$

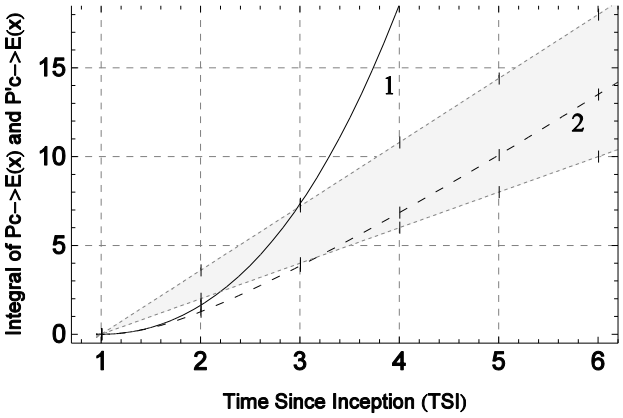
$$\begin{aligned}\int_1^c P'_{c\to E}(x) = &\quad \int_1^c \left(\frac{x}{c}\right)^c (-2-c+c^2+3x-x^2)\, dx \\ = &\quad -\frac{c(12-c(5+4c))+(\frac{1}{c})^c(-5+c(-2+c(3+c)))}{(2+c)(3+c)}\end{aligned} \tag{20}$$

6.2.8.3 Observations

We plot the integrals of the enhanced function, namely Eqs. 19 and 20 in Figure 27.



[Density distribution of (1) $P_{c\to e}(x)$, Eq. 3 and (2) $P'_{c\to e}(x)$, Eq. 13. The integral of (2) is given in Eq. 19.]



[Density Distribution of (1) $P_{c\to E}(x)$, Eq. 7 and (2) $P'_{c\to E}(x)$, Eq. 14. The integral of (2) is given in Eq. 20.]

Figure 28: Density distribution, in terms of the integrals of the enhanced versions of *ProbCache* for homogeneous and heterogeneous cache deployment scenaria. We see that in case of the enhanced version of *ProbCache* the density distribution of the values both for homogeneous and for heterogeneous cache deployments falls within the grey area.

Indeed, we see in Figure 27 that the density distribution of both versions of the enhanced *ProbCache* fall within the grey area. We argue that this is the ideal behaviour in terms of resource management and allocation in in-network caching environments. In Figure 27 and for line-plot (2), which represents the enhanced version of *ProbCache*, we see that all users have equal overall chances of caching their contents over the entire path, regardless of their distance from the source. Clearly, there is significant improvement compared to line-plot (1) in Figure 27, where the further away users are (in terms of hops from the source), the higher the chances they have to cache their contents at some point along the delivery path. The situation is similar in case of heterogeneous cache sizes as shown in Figure 27.

6.2.9 Summary of Theoretical Analysis and Findings

We have elaborated on the performance of the original version of *ProbCache* [43] and found that it favours content flows that connect far from the content source, over content flows that travel shorter distances. We verified this observation by capturing the value density distribution of

the basic equations of the algorithm (Eqs. 3 and 6), where we saw that longer content flows (in terms of hops from source) behave more aggressively and therefore, attempt to occupy resources along the entire delivery path (Figure 24). Based on these findings, we have modified the formula of $ProbCache$ in order for its value density distribution to evolve according to path lengths (Figure 25). Our theoretical findings show that the new version of $ProbCache$ (which we denote from now on as $ProbCache+$ and is given in Eqs. 13 and 14) is indeed more fair in terms of resource allocation and content flow multiplexing along a path of in-network caches (Figure 27).

We verify our theoretical findings presented above through extensive simulations in the next Section.

6.2.10 Performance Evaluation

6.2.10.1 Simulation Environment and Setup

We test our algorithm in a custom-built simulator, where we use *Least Recently Used* (LRU) caches. Given that the ultimate goal of *ProbCache* is to manage caching resources more efficiently, by reducing caching redundancy, the straightforward metric of interest is the *reduction of Server Hits*. Furthermore, the gain from serving user requests from intermediate caches, instead of travelling to the origin server, depends on the number of hops that the request travels before it eventually hits cached contents. Clearly, as the number of hops increases, the overall gain decreases. To measure this gain, we also monitor the *Hop Reduction Ratio*.

Apart from the above well-known metrics, which have been widely used in caching research in the past, we also introduce one extra metric in order to capture the *resource management and content flow multiplexing properties* of in-network algorithms. We call this metric *Content Multiplexing Fairness Index* (CMFI) and we define it as *the amount of caching resources that the algorithm under consideration has left unused in order for other content flows to exploit over the total amount of cache resources available along the entire delivery path*. The formula according to which the *CMFI* index is calculated is given below in Eq. 21 for the homogeneous and heterogeneous cache deployment cases, respectively:

$$CMFI_{c\to e} = \frac{\sum_1^x N_i}{\sum_1^c N_i} = \frac{xN_i}{cN_i} = \frac{x}{c} \text{and} CMFI_{c\to E} = \frac{\sum_1^x iN_i}{\sum_1^c iN_i} = \frac{x(x+1)}{c(c+1)} \quad (21)$$

The rationale behind the design of $CMFI$ is as follows: considering that a relatively realistic representation of the Internet topology comprises of fewer nodes at the core, which then fan out to more leaf/edge routers that finally reach out to residential connections, the *CMFI* index considers fairer to cache towards the edge of the network, rather than in the busy core. Therefore, the index gets bigger values for algorithms that tend to cache towards the edges of the network.

Although someone may argue that by caching contents at the very edge of a delivery path implicitly restricts access to those contents by clients connected further up towards the source, we consider that increased demand for content from across the delivery path will minimise the effect of this argument.

We use scale-free topologies following the Barabasi-Albert model [47], where nodes follow power-law degree distribution to reflect realistic Internet topologies. We use a benchmark topology of 200 nodes, but we note that the results remain qualitatively similar (*i.e.*, not in absolute numbers but in relative trends) in larger topologies of up to 700 nodes.

We set the exponent of the Zipf distribution of content popularity (a) to 1.2 to capture the case of medium-popularity content, while requests are generated following Poisson distribution. Again, content popularity affects the results in a quantitative way, as can also be verified from the results presented in our previous study in [43]. That is, although higher popularity (*i.e.*, exponent of Zipf distribution) results in higher performance for caching algorithms, this improvement is similar for all algorithms tested. Hence, qualitatively, the results follow similar trends for exponents between 0.5 – 1.5. As we show later in this section, when popularity exceeds a certain

threshold and causes small-scale *flash-crowd* events, then performance is affected more drastically.

The experiments run until 100,000 content requests have been successfully completed. We use two content servers, which are connected at the core of the network, that is, at the most well-connected part of our topology. This placement of servers was done on purpose in order to: *i)* reflect a relatively realistic network topology, and *ii)* avoid overwhelming isolated nodes (by attaching busy servers next to them), as this would (negatively) impact the behaviour of some of the protocols, as discussed further later on. We set the *cache-to-catalogue size* ratio to 0,0001% – 0,001% and as discussed before, we set the default cache size to be one second worth of traffic transmitted through the incoming link at each router.

We compare the performance of the following caching protocols:

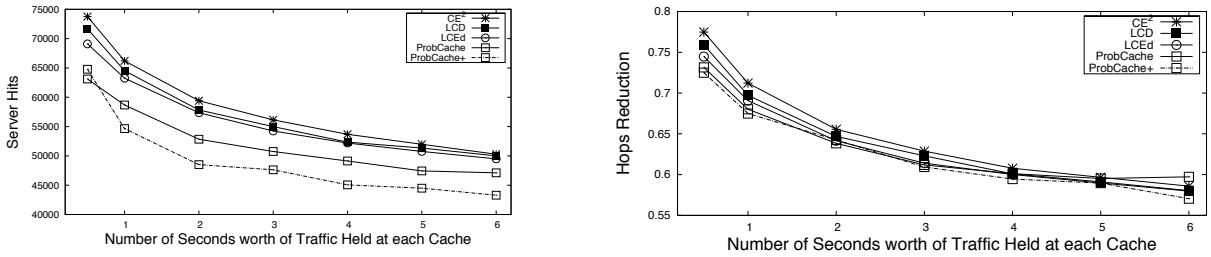
1. universal, or ubiquitous caching, a scheme that we call *Cache Everything Everywhere* (CE^2) (and was implicitly supported in [42]);
2. the *Leave Copy Down* (LCD) [44] algorithm proposed in the past for overlay caching topologies. According to LCD [20], cache hits cause contents to be copied one hop closer to the user, or one level down the cache hierarchy. ;
3. the *Leave Copy Edge* (LCEd) algorithm. LCEd caches contents deterministically one-hop before the client. According to the main design principle of $ProbCache$, contents should be cached closer to their destination with higher probability, in order to leave caching space at the core of the network for shorter content flows. Therefore, one might contend that a simpler algorithm that caches contents at the very edge of every content-flow path might achieve similar results. We, therefore, include LCEd in our performance evaluation for completeness;
4. the original version of $ProbCache$ (given in Eqs. 4), introduced in [43] and further elaborated in the previous sections;
5. the enhanced version of $ProbCache$ (given in Eqs. 7), which we introduced in the present study; we refer to the enhanced version of the algorithm as $ProbCache+$ from now on.

We note that in [43] we have evaluated the performance of $ProbCache$ against simpler approaches to in-network caching, *e.g.*, with probabilistic algorithms that cache with probability $p = 0.7$ and $p = 0.3$ at every cache. Our evaluation showed that such algorithms perform similarly to LCD, hence, we omit further comparison against such approaches in the present study.

We present three evaluation scenarios, which capture the most important aspects of the behaviour of the above-mentioned caching protocols. In the first scenario, we test the performance of the algorithm in an homogeneous cache-size environment, where all nodes in the network cache traffic for one second. In the second scenario, we use heterogeneous caches along the paths and in particular, we assume larger caches towards the edges of the network. Finally, we assess the performance of the caching protocols with regard to their convergence properties to popular content, that is, how fast are the protocols getting aware of a content that is becoming popular in order to keep copies of it in several caches and therefore, reduce server hits.

6.2.10.2 Scenario 1: Homogeneous Cache Environment

We evaluate the performance of the five caching protocols in a homogeneous cache-size deployment in the 200-node topology. In Figure 28, we present the overall performance of the five protocols. In terms of Server Hits, we see that $ProbCache$ outperforms the rest of the protocols by approximately 4-5%, while $ProbCache+$ reduces the number of Server Hits by an additional 5.5% compared to the original $ProbCache$ algorithm. This means an overall performance difference of approximately 10-11% for $ProbCache+$ compared to CE^2, LCD and LCEd, which perform largely the same.

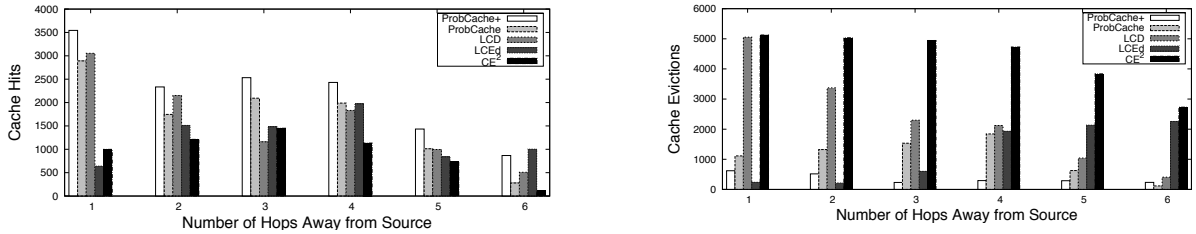


[Server Hits] [Hops Reduction]

Figure 29: Scenario 1: Homogeneous Cache Environment - Overall Results, Set of Simulations Applying Increasing Amount of Cache in each Experiment.

To get a better understanding of the performance of the protocols, we trace the per-hop *Cache Hits* and *Cache Evictions* of the protocols along a random 6-hop path in our topology. The results are presented in Figure 36. Figure 36 explains the overall behaviour of the protocols. For example, we see that LCD, in line with its core design principles, is pretty aggressive in evicting contents from caches close to the source of the content. In contrast, as we move away from the source LCD is becoming less aggressive. This behaviour results in more hits (and more evictions) at the core of the network (*i.e.*, close to the server) and less towards the edges. These hits, however, are not enough in order to guarantee satisfactory overall performance as shown in Figure 28. On the contrary, LCEd caches always at the edge of the path and hence, results in more aggressive behaviour (and therefore, more evictions) at the edges of the network. Both algorithms, however, always cache at least one copy of the content along the path; this action, however, might not always be affordable, given the limited caching resources available. Furthermore, LCD and LCEd cache at fixed and predefined places along the path, resulting in poor resource management and content distribution in the pool of available caching space.

The sophisticated resource management approach of *ProbCache* and *ProbCache+* clearly makes better use of available resources as shown in Figure 29. In particular, we see a clear difference between the cache evictions of the two versions of *ProbCache* and the rest of the protocols, which results in the overall better performance of the proposed protocol as shown in Figure 28. The resource management framework proposed here and integrated in both versions of the protocol caches incoming content chunks according to the resource availability along the path. This means that not all content chunks that traverse a path get necessarily cached in one of the router-caches, but instead, some chunks might not get cached at all. The decision of whether to cache or not is influenced by the *TimesIn* factor introduced earlier. We argue that this feature of the proposed protocol guarantees better overall resource management and utilisation and in turn, better network-wide performance.



[Per Hop Cache Hits] [Per Hop Cache Evictions]

Figure 30: Scenario 1: Cache hits and cache evictions along one delivery path.

Elaborating on the performance of the two versions of *ProbCache*, we observe the following. The original version of *ProbCache*, due to its slightly unfair content flow multiplexing behaviour (also shown in Figure 22) tends to cache contents half-way through the path from source to destination. This is clear from Figure 29, where nodes 2, 3 and 4 are forced to evict the highest

number of chunks. Although this does not necessarily translate to less cache hits, as shown in Figure 29 for nodes 2, 3 and 4, it does impact the overall performance of the protocol (see Figure 35). In contrast, the enhanced version of the algorithm (see Eqs. 14 and 15 and Figure 25), which targets more fair resource allocation along the path achieves less evictions in all nodes along the path, as shown in Figure 29. This verifies our claims in the previous section, where we presented the density distribution of the algorithm (see Figure 27), as well as the *local maxima* distribution (see Figure 27), where both plots fall within the *fair content distribution area* (grey area in Figure 27).

As a final step to evaluate the performance of the protocols and verify our theoretical findings regarding the multiplexing behaviour of the two versions of *ProbCache*, we plot the *Content Multiplexing Fairness Index* (CMFI) introduced earlier in this section. The result is shown in Figure 31.

We see that the design principle of LCD to gradually move contents down the cache-path towards the edge of the network results in low content multiplexing capability. In contrast, LCEd, by caching contents only at the edges of the network, leaves caching space for other content-flows. However, the deterministic nature of LCEd results in lower overall performance (*e.g.*, in terms of server hits) as we have shown in Figure 35. Finally, the original version of *ProbCache* shows little potential for fair content multiplexing, in contrast to our original design goals and in accordance to our findings in the previous Section. In contrast, *ProbCache* + exhibits ideal content multiplexing performance, by caching contents according to their path lengths, but also according to the path cache capability. *ProbCache* +, therefore, results in far better overall performance, both in terms of Server Hits and in terms of content multiplexing capabilities. These results verify the theoretical analysis of the two versions of *ProbCaches* in the previous Section, as well as claims regarding the performance of the rest of the protocols.

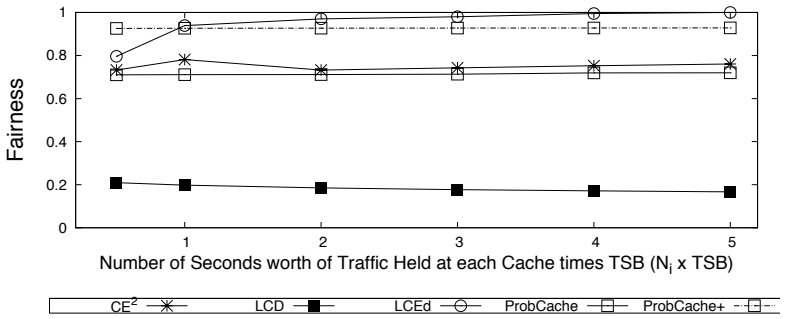


Figure 31: Scenario 1: Content Multiplexing Fairness Index (CMFI, see Eq. 22) for Homogeneous Cache Environments.

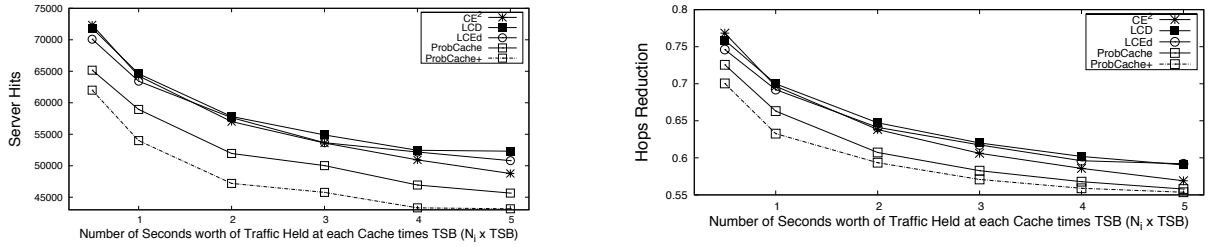
6.2.10.3 Scenario 2: Heterogeneous Cache Environment

We repeat the same experiment as in the previous scenario, but this time we deploy heterogeneous amounts of caches to routers along the path. In particular, as we move away from the server we increase the amount of traffic that can be temporarily stored in the routers' cache memory. We associate the cache capacity of each router with its distance from the server(s), according to the formula $N_i \cdot TSB$. Although this might be a slightly unrealistic setting for real deployments, due to the large number of servers or server farms deployed in relatively arbitrary locations, it gives us a good insight of the protocols' behaviour in heterogeneous cache size environments. We consider that the analysis of topological features (including cache sizing) is a subject of a different study, the initial results of which are presented in the next section.

The results of the heterogeneous cache size environment are presented in Figure 31. As we have also shown in [43], *ProbCache* becomes aware and exploits the extra cache resources along the path. The performance difference of *ProbCache* + in terms of Server Hits is now wider than

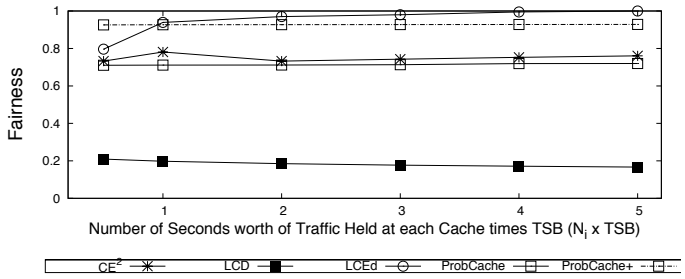
before and reaches to almost 13% compared to LCD, LCEd and CE^2 (see Figure 31). Furthermore and in contrast to the previous experiment, in the heterogeneous cache size setup, $ProbCache$ reduces significantly the Hop Reduction Ratio, as can be seen in Figure 31. The difference of $ProbCache+$ is in the order of 7%, compared to LCD, LCEd and CE^2, which is rather significant, if we consider the topology characteristics (*i.e.*, mean valence of the degree in the topology is 2, hence, very few paths are longer than 6-7 hops).

Finally, in terms of *content multiplexing fairness*, we observe in Figure 31 that the fairness performance of the original version of $ProbCache$ has now increased. This is a somewhat expected result, if we consider that the amount of cache increases as we move towards the middle and the edge of a given path, where $ProbCache$ gets its highest value and tends to cache contents (see also Figure 22). On the other hand, $ProbCache+$ behaves as expected; it pushes contents towards the edges of the paths, while at the same time it manages resources in an efficient way and results in more contents being cached along the path. As we show in the next scenario, although probabilistic caching might not cache potential popular content immediately, the design of $ProbCache$ converges to the identification (and therefore, caching) of popular content almost as fast as CE^2.



[Server Hits]

[Hops Reduction]



[Content Multiplexing Fairness Index (CMFI, see Eq. 21) for Heterogeneous Cache Environments]

Figure 32: Scenario 2: Heterogeneous Cache Environment - Overall Results, Set of Simulations Applying Increasing Amount of Cache in each Experiment. Cache size is set according to the formula $N_i \cdot TSB$, where TSB is the Time Since Birth value.

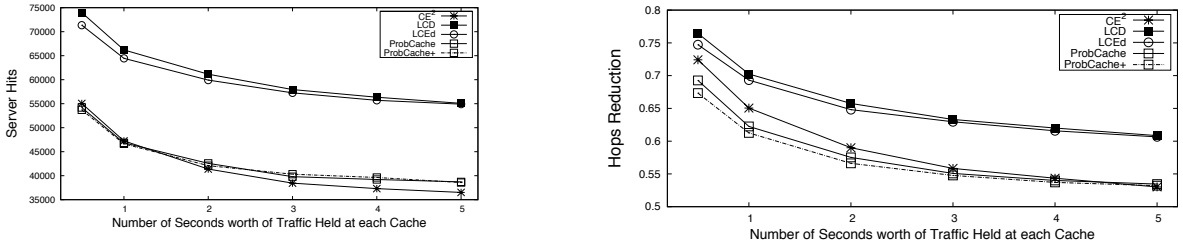
6.2.10.4 Scenario 3: Caching Convergence to Popular Content

In our final evaluation scenario, we test the performance of the caching protocols with regard to their convergence time in case of an emerging popular content. That is, we consider that after the 1000 th second of the simulation, a previously unpopular content, which we refer to as the *Content of Interest* or *CoI*, becomes popular and constantly receives 5% of the total requests generated per second. We use the same 200-node topology and the homogeneous cache size setup to capture the behaviour of the algorithms in the basic setting. We report that in case of heterogeneous caches, results follow similar trends. The results are summarised in Figure 32 and Figure 33.

In Figure 32, we see significant performance differences between $ProbCache$, $ProbCache+$, CE^2 (which perform roughly the same) and LCD, LCEd, both in terms of Server Hit ratio (which now is very close to 20%, Figure 32) and in terms of Hop Reduction Ratio (which is approximately 10%, Figure 32). In particular, LCD and LCEd being deterministic algorithms that cache one copy

of every content in specific points along the content delivery path fail to exploit the popularity of the *CoI*. This owes to the fact that the deterministic nature of caching in these cases results in contents (and particularly the *CoI*) being evicted from the cache before it receives further requests. CE^2, which also caches deterministically escapes (to a certain extend) this behaviour due its inherent caching redundancy feature.

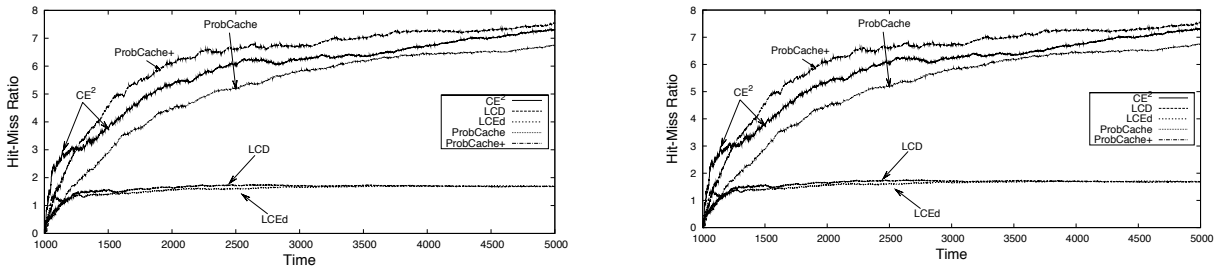
Although someone might expect that caching contents probabilistically might result in slow convergence and therefore, reduced performance, both versions of $ProbCache$ invalidate this claim. The sophisticated design of $ProbCache$ (and $ProbCache+$) bases its probabilistic caching behaviour on the amount of traffic served by the router *per unit time* and not based on arbitrary assumptions regarding the server catalogue size. Having said that, $ProbCache$ is essentially choosing contents from a small range of incoming traffic (see Eqs. 4, 5 and 6). This feature of the proposed algorithm reduces the caching redundancy and increases the number of contents cached along the entire path. Therefore, although popular content might not get immediately cached in all routers along the path, it still gets cached at one point along the delivery path and therefore avoids being fetched from the origin server.



[Server Hits] [Hop Reduction Ratio]

Figure 33: Scenario 3: Caching Convergence to Popular Content - Overall Results, Set of Simulations Applying Increasing Amount of Cache in each Experiment.

To prove our claims further, we plot *the number of caches that hold the CoI* in Figure 33 and the *hit-miss* ratio in Figure 33 as the experiment progresses. In Figure 33, we observe that indeed CE^2 populates the caches faster with the *CoI* compared to $ProbCache$ and $ProbCache+$, but this difference is in the order of 120 to 150 seconds, as can be seen in the beginning of the experiment from the 1050 th to the 1200 th second. In the long term, however, we see that CE^2 evicts the *CoI* faster than new requests come in and therefore results in less cache hits. This is shown in Figure 33, where although for the interval between the 1050 th and the 1200 th second CE^2 has higher *hit-miss* ratio, after the 1200 th $ProbCache+$ is performing better. With regard to LCD and LCEd, we see in Figure 33 that the deterministic approach to content caching results in very poor performance in terms of convergence to caching of popular content. This behaviour is due to high contention for caching slots in the fixed places where these algorithms cache contents. In turn, this results in evicting the *CoI* faster than new requests come in.



[Number of Caches Holding *CoI*] [*Hit-Miss* Ratio]

Figure 34: Scenario 3: Caching Convergence to Popular Content - Further Results.

6.2.10.5 Summary of Results

We have evaluated the performance of five different caching protocols with regard to their resource management and allocation properties. Our evaluations included a variety of different network settings, traffic conditions and patterns, as well as various performance metrics. We conclude that deterministically caching content chunks in fixed and pre-determined places along a delivery path results in poor resource management and in turn, reduced overall network performance. In contrast, sophisticated algorithms that take into account, or approximate, the amount of available cache resources along the delivery path have the potential to improve both network utilisation and user-perceived quality of service. This was demonstrated during our performance evaluation presented in the previous subsections by:

• significant reduction in server hits, which in turn, reduces the delivery time, and

• huge reduction in the number of cache evictions (with parallel increase of cache hits) in case of $ProbCache$, which translates to reduction of traffic redundancy and in turn, bandwidth consumption.

Finally, caching at carefully selected places in the network does not degrade the convergence performance of caching protocols in case of small scale flash crowd events, as we demonstrated in our last scenario. In contrast, we have shown that $ProbCache$ performs almost similar and sometimes even better than universal caching, which is clearly the best candidate when it comes to flash crowd events. In case of $ProbCache$ this is due to sizing content caches according to the amount of traffic that they serve per unit time, which results in identification of popular contents much faster than we initially expected.

6.3 Centrality-based In-network Caching

The concept and algorithm of the centrality-based in-network caching has been described in D4.2 [4] and published in [45] and [46].

In this deliverable, we proceed to provide the full evaluation of the approach.

6.3.1 Model Description

Let $G = (V, E)$ be an undirected network with $V = \{v_1, \ldots, v_N\}$ nodes and $E = \{e_1, \ldots, e_M\}$ links. We denote $F = \{f_1, \ldots, f_R\}$ the content population in the system and $S = \{s_1, \ldots, s_P\}$ the set of content servers, each associated to a $v \in V$. The content population is randomly hosted in S and we assume that each content object is hosted permanently in only one server.

Content requests are assumed to arrive in the network exogenously and the content request arrival process for content unit $r, 1 \leq r \leq R$, follows the Poisson process with mean rate, $\lambda = \sum_{r=1}^{R} \lambda_r$, whereby λ_r is the rate of exogenous content request for f_r. A *cache hit* is recorded for a request finding a matching content along the content delivery path. Otherwise, a *cache miss* is recorded. In the event of a cache miss, the content request traverses the full content delivery path to the content server. Following the convention in the literature, we assume that content units are of the same size and each cache slot in a cache store can accommodate one content unit at any given time. When a cache store is full, the least recently used content will be discarded in the event of an arrival of a new uncached content.

The objectives of this study are: (1) to examine the caching performance of such a system under different caching schemes, (2) to gain insights into the behavior of ubiquitous caching and (3) to develop and understand more sophisticated caching algorithms for achieving better gain.

In this evaluation study, we compare our centrality-based caching schemes against the one defined in [42] and a simple and naïve random caching scheme. Specifically, we have the following caching schemes:

- A ubiquitous indiscriminate caching scheme similar to that defined in [42]. Hereafter, this scheme is denoted as *CCN*.

- Our caching scheme based on the concept of betweenness (cf. D4.2 [4]). Hereafter, this scheme is denoted as *Betw*.
- Our caching scheme based on the concept of ego network betweenness (cf. D4.2 [4]). Hereafter, this scheme is denoted as *EgoBetw*.
- A naive random caching strategy which simply caches randomly at only one intermediate node along the delivery path per request, using LRU cache eviction policy. Hereafter, this scheme is denoted as *Rdm*.

6.3.2 Performance Metrics and Evaluation Methodology

We use a custom-built simulator for the study of the dynamics of content caching in order to evaluate our proposal. All nodes in the simulator are cache-enabled and we perform the experiments based on the specifications described above for the different caching schemes.

Caching in networks aims to: (1) lower the content delivery latency whereby a cached content near the client can be fetched faster than from the server, (2) reduce traffic and congestion since content traverses fewer links when there is a cache hit and (3) alleviate server load as every cache hit means serving one less request. We use the *hop reduction ratio*, β as the metric to assess the effect of the different caching schemes on (1) and (2) above while we use the *server hit reduction ratio*, γ on (3).

$$\text{Hop reduction ratio}, \beta(t) = \frac{\sum_{r=1}^{R} h_r(t)}{\sum_{r=1}^{R} H_r(t)} \tag{22}$$

where $H_r(t)$ is the path length (in hop count) from client(s) to server(s) requesting f_r from time t-1 to t and $h_r(t)$ is the hop count from the content client to the first node where a cache hit occurs for f_r from t-1 to t. If no matching cache is found along the path to the server, then $h_r = H_r$. In other words, the hop reduction ratio counts the percentage of the path length to the server used to hit the content given caching in intermediate nodes. In a non-caching system, $\beta = 1.0$.

$$\text{Server hit reduction ratio}, \gamma(t) = \frac{\sum_{r=1}^{R} w_r(t)}{\sum_{r=1}^{R} W_r(t)} \tag{23}$$

where $W_r(t)$ is the number of requests for f_r from t-1 to t and $w_r(t)$ is the number of server hits for f_r from time t-1 to t. Note that high hop reduction does not directly translate to high server hit reduction.

We seek to draw insights from the inspection of network topologies with very different structural properties - (1) *k-ary* trees which have almost strict regular structure (i.e., all nodes besides the root and leaves have the same k+1 degree) and (2) scale-free topologies following the Barabasi-Albert (B-A) power law model [47] which accounts for the preferential attachment property of the Internet topology and results in graphs with highly skewed degree distribution. It is interesting to note that the betweenness distribution of B-A graphs also follows the power law model [48].

Content requests for different content are generated based on Zipf-distribution with $\sum_{r=1}^{\mathrm{R}} \left(\frac{\mathrm{C}}{r^{\alpha}}\right) = 1$ where the probability for a request for the r^{th} popular content is C/r^{α} with α being the popularity factor. We use α =1.0 and requests originate randomly from all nodes[3]. Each simulation run begins with all cache stores being empty (i.e., cold start). The content population is randomly distributed in the network with each content object being hosted persistently in one server. Unless otherwise specified, the simulations are run with the following parameters: total simulation time = 200 s, λ = 5,000 request/s and uniform cache store size = 100 content.

[3] From our results, we note that the order of performance amongst the caching schemes remains unchanged for $0.6 \leq \alpha \leq 1.5$. So, the results presented here are valid for these values of α.

6.3.3 A Preliminary Test

The ubiquitous indiscriminate caching scheme has already raised doubts (e.g., [50]). In the general cache-related literature, some authors have already questioned this aggressive *cache-everything-everywhere* strategy [44][51] [52]. The basic reasoning is that since the caching capacity is usually much smaller than the overall population of the items to be cached, it has the property of high cache replacement error. We run a preliminary experiment to illustrate this property of ubiquitous caching with a motivating example. We compare the two caching schemes in a 7-node string topology where $P = 1$, s_1, is located at v_1 (root) while content requests originate exogenously from other nodes. We provide the detailed simulation setup later in this section. We observe, in Figure 35, that even random caching at just a single node along the content delivery path can reduce both the number of hops required to hit the content and the server hits in comparison to ubiquitous caching (*CCN*).

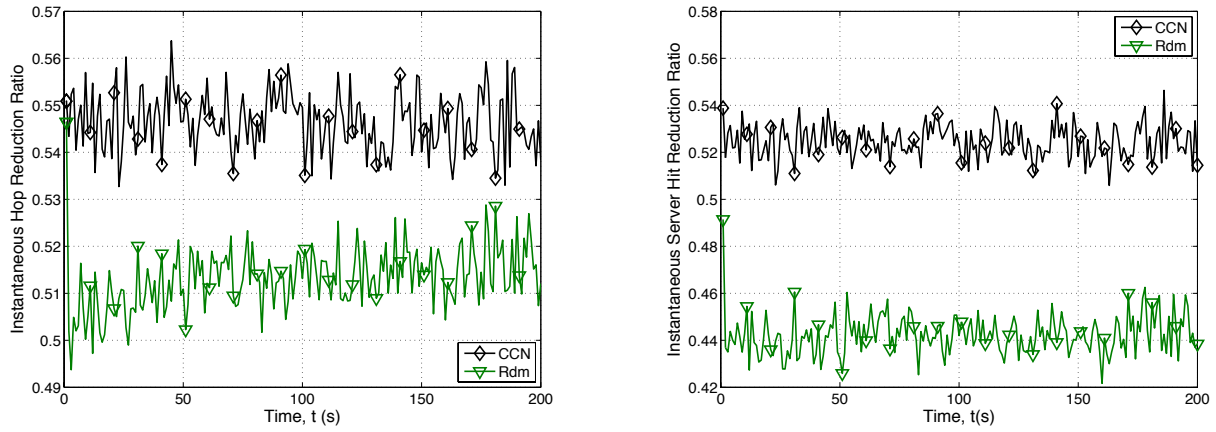


Figure 35: Simple random caching outperforming ubiquitous caching in the number of hops to hit the content (left) and reduced server hits (right).

Based on the above observations, we realize that caching indiscriminately does not necessarily guarantee the highest cache hit rate. Intuitively, this may be caused by the high cache replacement rate on non-selective caching schemes such as *CCN*. By caching indiscriminately, a content in a cache is more likely to be replaced before it gets a hit. On the other hand, this result cannot be used as conclusive evidence that caching less is better since the string topology constrains to quite a large extent the diversity of the content delivery paths (i.e., all delivery paths are fully or partially overlapping), a fact that indirectly increases the probability of a cache hit.

6.3.4 Performance in *k*-ary Tree Topologies

6.3.4.1 Instantaneous Behavior

A *k-ary* tree is defined via two parameters, namely k, the spread factor, denoting the number of children each node has and D is the depth of the tree from root. We show in Figure 36 the instantaneous behavior of the different caching schemes for both β and γ in a 5-level binary tree ($k = 2, D = 4$). All caching schemes reach a stationary performance after a few seconds. We point out that since all simulations go through a *warm-up* phase, *CCN* always reaches the stable performance level first. This is due to its "*always cache*" policy.

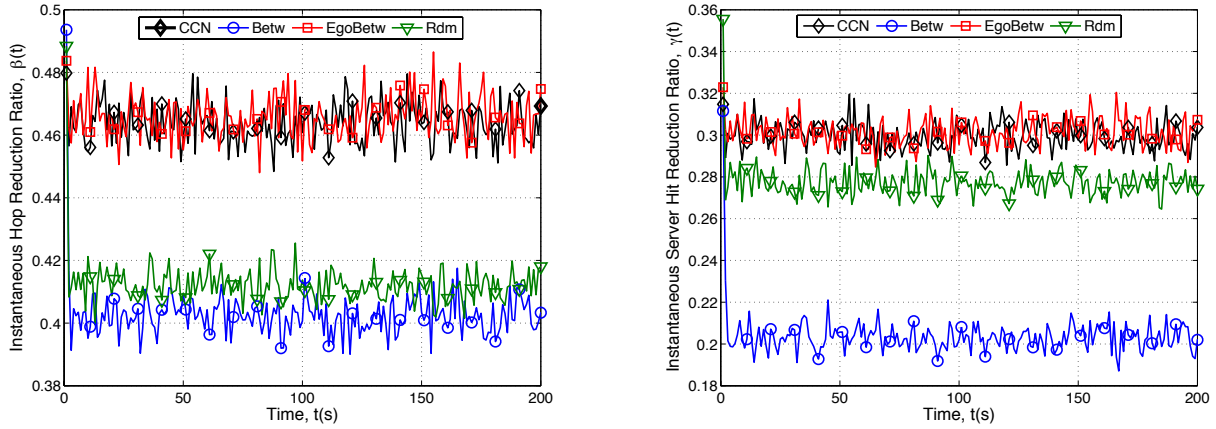


Figure 36: Instantaneous behaviour of the caching schemes for a binary tree; (left) β, (right) γ.

We observe that both *Betw* and *Rdm* perform better than *CCN* for both metrics. Tracking the evolution of the cache stores over time revealed that this is due to the high cache replacement rate in *CCN*. Replacing cached content rapidly causes content often being evicted before the next matching request is received. The effect is magnified considering that the whole chain of caches on the delivery path is affected. This is the fundamental basis on why the counter-intuitive "*less for more*" caching scheme proposed here can be true. We further observe that the argument that caching selectively may increase cache miss is untrue in *k-ary* trees. We do find that there are more cache misses if the caching node is randomly selected rather than caching at nodes with high betweenness. Finally, an interesting observation is that instead of approximating the performance of the *Betw* scheme as it was meant to be, *EgoBetw* actually performs at the same level as *CCN*. This is due to the regularity of the topology whereby nodes between the root and the leaves have the same ego network and thus, have the same C_B. Since the algorithm specifies that all nodes with equal highest C_B along the delivery path should cache, in this case EgoBetw is simply reduced to a similar behavior with *CCN*.

6.3.4.2 Effect of Topology Features on Performance

In *k-ary* trees, D affects the expected path lengths and k impacts the path diversity. We now study the validity of the previous observations in different configurations of *k-ary* trees by obtaining the β at 95% confidence interval for a range of depths and spread factors. Our results in Figure 37 suggest that the caching schemes exhibit consistent behavior for different *k-ary* trees.

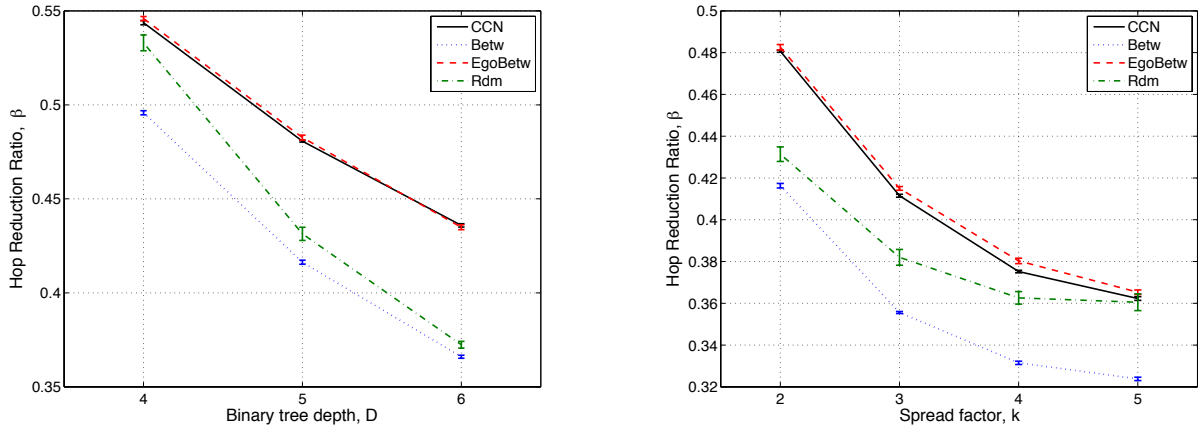


Figure 37: *Betw* consistently outperforms the rest over different D (left) and k (right).

We find that while the performance distance between *CCN* and *Betw* remains approximately constant, *Rdm* does not exhibit such consistency. In general, *Rdm* always has the highest variance

due to the randomness implicit to the algorithm. *Rdm* performs increasingly better in terms of hops saved when D is increased and k is decreased. This is due to the fact that each node has equal probability to cache content and in effect, distributes cache replacement operation uniformly across different nodes. In turn, this results in content being cached longer when compared to *CCN*. It increases the cache hit probability especially in topologies with very low number of content delivery paths. This, however, is counter-balanced by the increased number of branches in the topology, whereby a greater number of cache misses will occur. Our *Betw* scheme does not suffer from such a drawback since the caching node always has the highest probability of getting a cache hit and thus maintains stable cache hit (reducing server hits) and network resource gain (reducing the content delivery hop count).

6.3.5 Performance in Scale-free Topologies

6.3.5.1 Instantaneous Behavior

Although regular graphs lend themselves to tractability in modeling, real-world Internet topologies are not regular but follow a power law degree distribution [47]. As such, we consider scale-free topologies following the construction method described in [47] (referred to as B-A graphs hereafter). We show in Figure 38 the performance of the different caching schemes in a B-A graph with N=100 over time. First and foremost, we see that the performance of both our centrality-based caching schemes (*Betw* and *EgoBetw*) perform better than *CCN* for both metrics and *EgoBetw* now approximates closely *Betw*. This is because, without the regular structure, the ego networks of the nodes within the B-A graphs reflect correctly their actual betweenness. This result, thus, suggests that the more scalable and distributed *EgoBetw* algorithm can be used for irregular graphs.

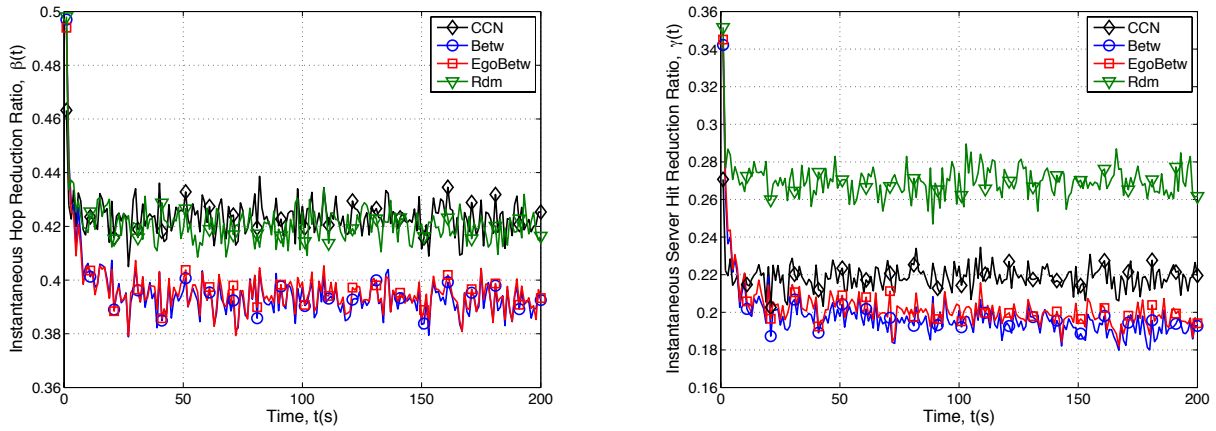


Figure 38: Instantaneous behaviour of the caching schemes for a B-A graph; (left) β, (right) γ.

Second, we observe that *Rdm* no longer outperforms *CCN*. In fact, it performs at the same level as *CCN* with respect to hop reduction and due to the highly skewed degree distribution in the topology, it fails to alleviate load from the server (i.e., it has the highest number of cache misses).

6.3.5.2 Effect of Topology Features on Performance

Unlike *k-ary* trees which are fully described via the tuple (k, D), each generation of a B-A graph with the same parameters results in a different topology since the links are created based on the probability proportional to the attractiveness of existing nodes (i.e., preferential attachment). We evaluate the caching schemes over 50 B-A graphs with N=100 and mean degree = 2. From Figure 39, both centrality-based caching schemes always perform better than the rest. The mean β achieved for *CCN*, *Betw*, *EgoBetw* and *Rdm* are 0.47581, 0.44583, 0.44845 and 0.47891 respectively. The variances obtained are 2.86357×10^{-4} (*CCN*), 4.91978×10^{-4} (*Betw*), 4.83752×10^{-4} (*EgoBetw*) and 8.83494×10^{-4} (*Rdm*). As expected, *Rdm* has the highest variance. *Rdm* is worse

than *CCN* in many cases even with the topology having the same properties. This is due to the skewed node degree distribution of the graph that increases the probability of the scheme caching at nodes having low cache hit probability. Figure 40 shows how ego network betweenness approximates betweenness in a B-A graph.

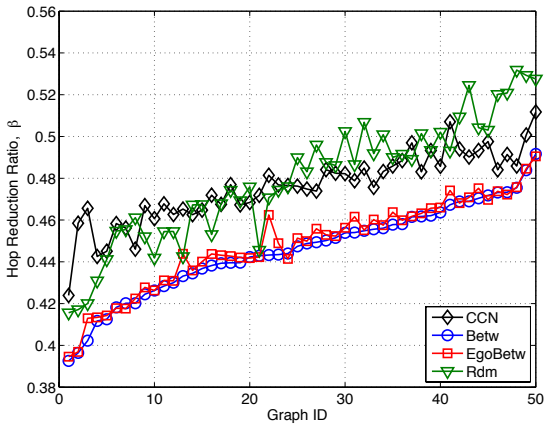


Figure 39: Performance with different B-A graphs (N=100).

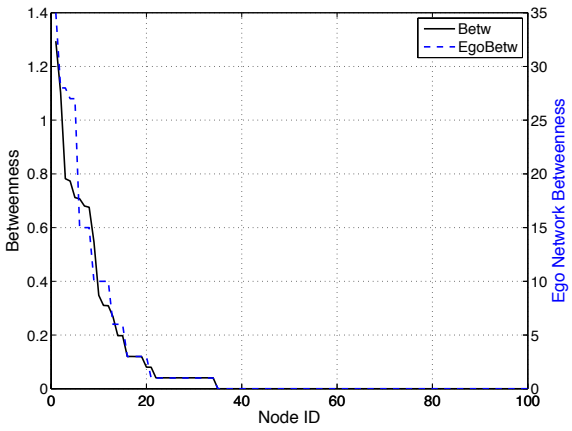


Figure 40: A sample ego network betweenness and betweenness values of the nodes in a B-A graph.

From Figure 41 (left), we observe again that centrality-based caching schemes provide the best hop reduction ratio while *Rdm* exhibits inconsistent gain across B-A graphs with different sizes. We observe that as the size of the topology increases, *Rdm* gradually performs worse than *CCN*. The power-law distribution of betweenness in B-A graphs plays a vital role in this phenomenon as it results in high number of nodes having low probability of getting a cache hit. Since *Rdm* does not differentiate the centrality of the nodes, there is higher probability of *Rdm* caching at these "*unimportant*" nodes. Note that this observation is untrue for *k-ary* trees (the case when D is increased) due to the high number of overlapping shortest paths (an obvious example being the string topology).

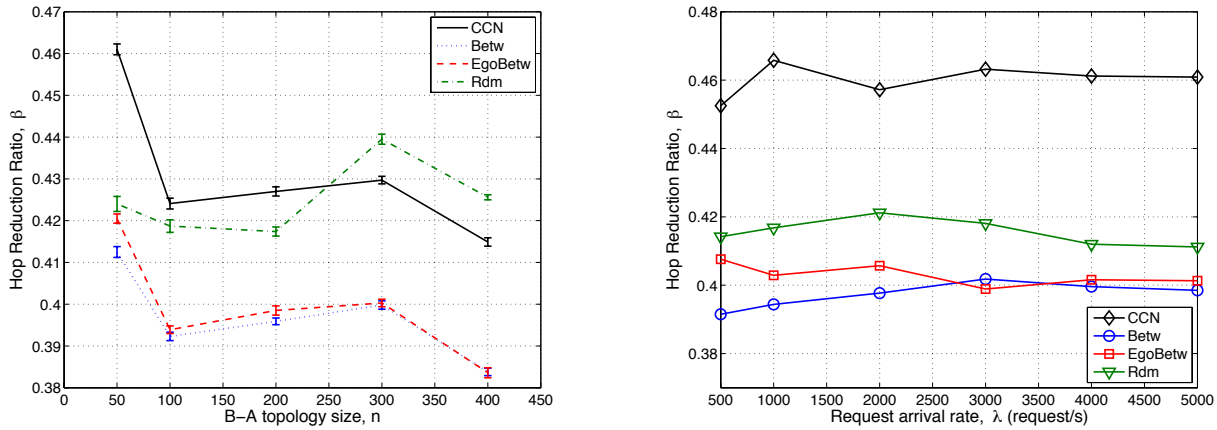


Figure 41: Hop reduction ratio for different B-A graph sizes (left) and request rate, λ (right).

In Table 18, we provide representative results of the different caching schemes across the different topologies in terms of number of hops and server hits saved. It is clear that *Betw* reliably achieves better gains (both in terms of hop and server hit reduction) in comparison to *CCN*. For instance, it reduces server hits over 30% and hop count over 17% in comparison to *CCN* in the string topology.

Table 18: Sample performance achieved after 200s in different types of topology.

Caching Scheme	String (D = 10, k=1)		k-ary Tree (D = 4; k = 2)		B-A (N = 100)	
	$\sum h$	$\sum w$	$\sum h$	$\sum w$	$\sum h$	$\sum w$
CCN	2,6839,45	498,603	2,684,325	299,657	2,137,015	211,852
Betw	2,211,248	337,362	2,331,061	203,673	2,045,852	204,479
EgoBetw	2,680,614	497,146	2,698,153	301,797	2,074,089	207,628
Rdm	2,206,002	377,289	2,386,569	277,575	2,195,303	291,560

6.3.6 Performance in Real AS-level Topologies

To further verify our findings, we proceed to assess the caching performance of the different caching schemes in a real-world Internet topology. We focus on a large domain-level topology, extracting a sub-topology from the CAIDA dataset [14]. The topology is rooted at a tier-1 ISP (AS7018) and contains 6,804 domains and 10,205 links. We do not aggregate stub domains while sibling domains / links are not considered. In a similar manner to the previous simulation setup, all content servers and clients are randomly distributed across the topology. Figure 42 shows both the hop reduction and server hit reduction ratios achieved in this setup.

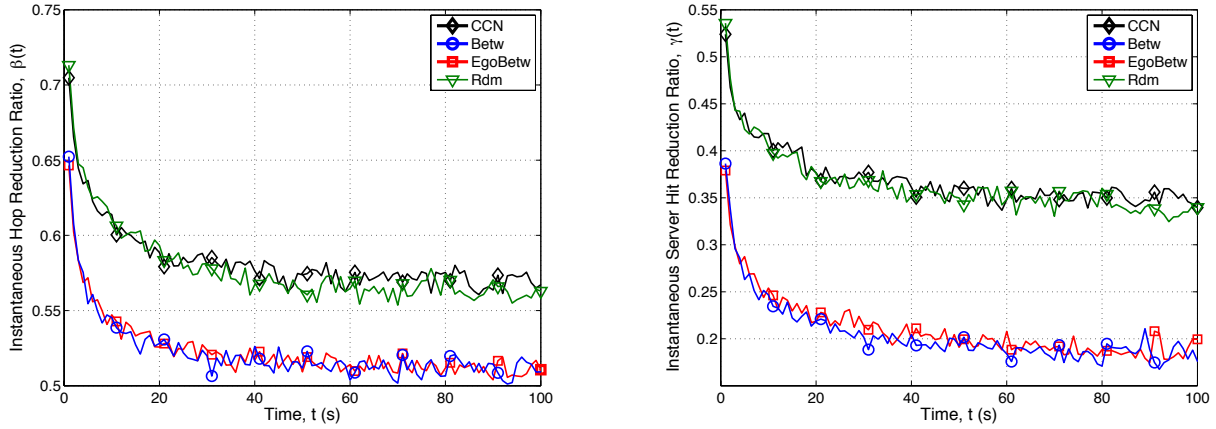


Figure 42: Instantaneous behaviour of the caching schemes in a large-scale real Internet topology; (left) β, (right) γ.

The results show that the different caching schemes behave in a similar fashion to the B-A graphs but not to *k-ary* trees, reinforcing the notion that B-A graphs reflect better real network topologies. These results further confirm the validity of our centrality-based caching scheme even in large-scale real network topologies. We provide in Table 19 the mean and variance of β and γ at stable operation phase (i.e., after the initial transient phase).

Table 19: The mean and variance of the performance achieved after the initial transient phase in different types of topology.

Metric	Caching Scheme	*k*-ary Tree (D = 4, k=2)		*B-A* (N=100)		CAIDA (N = 6,804)	
		Mean	Variance	Mean	Variance	Mean	Variance
β	*CCN*	0.46346	4.24717×10^{-5}	0.42315	3.01174×10^{-5}	0.56849	3.31567×10^{-5}
	Betw	0.40216	2.78626×10^{-5}	0.39235	2.58357×10^{-5}	0.51832	3.55136×10^{-5}
	EgoBetw	0.46580	4.64435×10^{-5}	0.39390	2.5936×10^{-5}	0.51605	2.81248×10^{-5}
	Rdm	0.41187	2.044×10^{-5}	0.41981	2.59586×10^{-5}	0.55843	1.90826×10^{-5}
γ	*CCN*	0.29924	4.36482×10^{-5}	0.21869	3.2002×10^{-5}	0.35137	4.67507×10^{-5}
	Betw	0.20298	2.89143×10^{-5}	0.19390	3.24316×10^{-5}	0.19005	1.20289×10^{-4}
	EgoBetw	0.30121	4.64179×10^{-5}	0.19877	3.02138×10^{-5}	0.18697	1.14169×10^{-4}
	Rdm	0.27729	3.00353×10^{-5}	0.26913	5.30707×10^{-5}	0.33394	1.28169×10^{-4}

6.3.7 Caching Operation Overhead

From the above, we have shown the consistent gain of *(Ego)Betw* in terms of cache hit and server hit. In this section, we provide an illustration on how much less caching is done to achieve this gain. We measure the number of caching operations (i.e., the actual process of storing and evicting content in the nodes). Figure 43 shows the recorded number of caching operations for the different caching schemes in a *k-ary* tree with $k = 2$ and $D = 5$ and a B-A graph with $N = 100$.

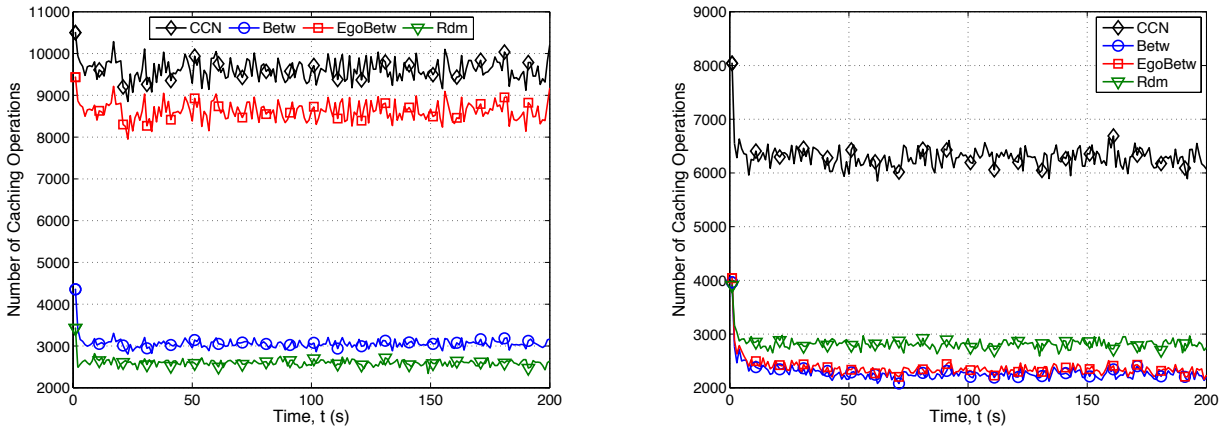


Figure 43: Number of caching operations for the different caching schemes; (left) k-ary tree, (right) B-A graph.

CCN always has the highest number of caching operations compared to the others since it caches non-selectively in every node along a content delivery path. Also, it is apparent that the caching operation of *CCN* is highly dependent on the length of the content delivery paths. We tracked the path length of each content request between the content server and user and found that the average path lengths are 5.78 hops and 2.87 hops for the binary tree and B-A graph respectively. Note that the B-A graph has smaller average path length albeit having higher number of nodes. This is due to the existent of highly connected hubs (i.e., small-world effect) in B-A graphs.

Betw, however, cache approximately 69% and 64% less than what *CCN* needed in *k-ary* tree and B-A graph respectively. *EgoBetw* exhibits similar behavior in B-A graph but not in *k-ary* tree. The reason for this is simply because all the nodes that are not root or leaf nodes in the tree have the same C_B values and thus, will all cache.

6.4 Conclusions

We have argued that caching named chunks in network routers' DRAM memory, as opposed to caching large objects or files in proxy disks, calls for reconsideration of past approaches to caching. In-network caching in ICNs has to happen in an uncoordinated and distributed fashion, taking also into account the available cache resources. We have therefore, introduced the concept of *resource management* for in-network caching environments and have argued that indiscriminate and/or deterministic caching presents little potential for efficient resource utilisation.

We have proposed two algorithms for in-network caching within the context of the COMET system. The first, ProbCache, is an algorithm that approximates the capability of paths to cache contents, based on path lengths, and multiplexes content flows accordingly. The ultimate goal of ProbCache is to utilise resources efficiently, reduce caching redundancy and in turn, network traffic redundancy. We have considered both homogeneous and heterogeneous cache sizes and have adjusted ProbCache to fit in both environments.

We report savings of up to 20% in server hits; 7-8% in the number of hops to hit cached contents; and reduction by an order of magnitude in cache evictions, which directly translates to network traffic redundancy elimination by the same proportion.

The second algorithm proposed here is a centrality-based in-network caching approach. We first demonstrated that a simple random caching strategy (*Rdm*) can outperform (though inconsistently) the current pervasive caching paradigm under the conditions that the network topology has low number of distinct content delivery paths and high average delivery path length. Our proposed in-network caching strategy based on the concept of betweenness centrality (*Betw*) such that content is only cached at the nodes having the highest probability of getting a cache hit along the content delivery path. Our design ensures the content always spreads towards content

users and thus reduces the content access latency. We also proposed an approximation of it (*EgoBetw*) for scalable and distributed realization in dynamic network environments where the full topology cannot be known *a priori*.

We compared the performance of our proposals against the ubiquitous caching of the *CCN* proposal [42] (*CCN*). Based on our extensive simulations, we observed that *Betw* consistently achieves the best hop and server reduction ratios across topologies having different structural properties without being restricted by the operating conditions required by *Rdm*. Our results further suggest that *EgoBetw* approximates closely *Betw* in non-regular topologies (e.g., B-A graphs) and thus presents itself as a practical candidate for the deployment of this approach. Besides synthetic topologies (i.e., *k-ary* trees and B-A graphs), the observations were further verified through a large-scale real Internet topology. We also showed that the caching overhead of *CCN* is more than 60% higher than our *Betw*. Thus, we conclude that indeed caching less can actually achieve more and that our proposed *(Ego)Betw* approach is a potential candidate for realizing this promise.

7 Performance of multi-constrain and multipath routing

7.1 Mullti-constraint and multipath routing

Current Internet relies on BGP-4 shortest path routing protocol, which establish a single routing path between any two domains. Such approach limits the effectiveness of content delivery because: (1) the network delivers the content regardless of its transfer requirements, what generally leads to degradation of the quality experienced by consumers, and (2) downloading of popular content may provoke network congestion, since single routing path going from content server's to customers' domains may become congested.

Exploiting the fact that COMET defines a new network architecture, we may go beyond the above limitations and define multi-constraints and multi-path routing protocol, which is performed by Routing Awareness Entity (RAE), [4], [6]. Multi-constraints routing selects paths that satisfy a given set of constraints [57], [58]. Let us consider the network as a directed graph $G(N, E)$, where N represents the set of domains, while E is the set of links. Each link $u \rightarrow v$, $u,v \in N$, $u \rightarrow v \in E$, is characterized by m-dimensional vector of non-negative link weights $w(u \rightarrow v) = [w_1, w_2, ..., w_m]$. *Any* path p between two nodes is characterized by a vector of path weights $w(p)=[w_1^{(p)},w_2^{(p)},...,w_m^{(p)}]$, where $w_i^{(p)}$ is calculated as a concatenation of the link weights w_i of each link belonging to path p. The multi-constraints routing finds a set of feasible paths, $f \in F$, going from server domain to consumer domain. The path is feasible if its path weights satisfy the assumed constraints: $w_i^{(f)} < l_i, i=1, ..., m$, where L is given vector of constraints $L=[l_1, l_2, ..., l_m]$.

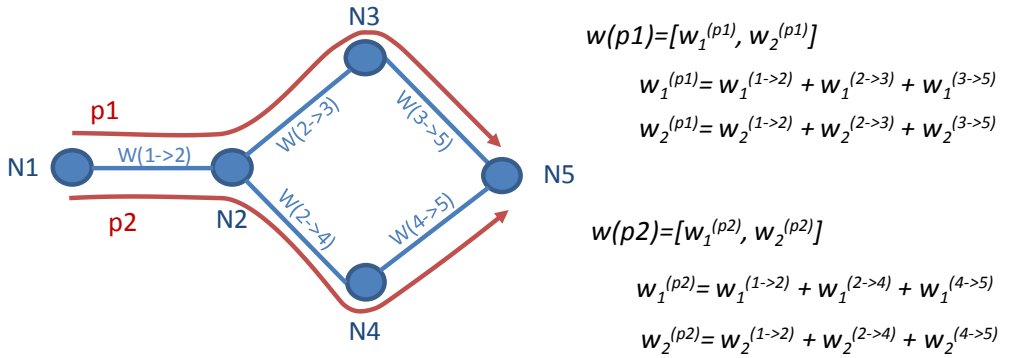


Figure 44: The illustration of multipath routing with two paths between domains N1 and N5.

Figure 44 presents an example on how RAE establishes paths. Domain N1 has two alternative paths (i.e., p1 and p2) going towards domain N5. Each path is characterised by its two-dimensional vector of weights. The path weights are calculated as a concatenation of links weights.

The multi-constraints routing belongs to the class of multi-criteria optimization problems that are in principle NP-complete. Although different methods for multi-criteria optimization have been proposed, they cannot be directly used in routing protocols because most of them assume independent decision makers operating on its own decision space. On the other hand, routing protocols create paths in distributed manner where decision space of a given domain is determined by information propagated by neighbouring domains. Consequently, the end-to-end path is a result of the sequence of decisions taken by the involved domains based on their knowledge about paths, local preferences and assumed constraints (local decisions in path vector routing protocol).

The proposed multi-constraints and multi-path routing protocol creates a set of content delivery paths between server and consumer domains with respect to content QoS requirements. RAE follows the path vector principle, where each domain advertises its preferred paths to its neighbours. Each path is described by a list of ASes and the corresponding vector of path weights $w(p)$ calculated as concatenation of links weights $w(u \rightarrow v)$. These characteristics allow RAE to eliminate routing loops and remove unfeasible paths. Furthermore, RAE removes dominated

paths, i.e., paths for which exists another path with all weights $w_i^{(p)}$ lower than weights of paths in question. Remaining paths form a set of preferred paths. In order to achieve the optimum solution, all preferred paths should be advertised to neighbouring domains. However, for scalability reasons, the number of advertised paths must be limited to some reasonable value. Based on performed experiments, we recommend limiting the number of advertised paths to 3 ÷ 5 paths. The smaller value limits the gain achieved by multipath routing. On the other hand, dissemination of more than 5 paths only slightly improve effectiveness but significantly increases the size of routing tables. The key problem in the proposed heuristics is how to choose the right paths that would be advertised to peering domains. We believe that selection algorithms should prefer paths with larger distance between weight *w(f)* and constraint L. In this way, adjacent domains receiving those paths will have a greater chance of finding feasible paths. Therefore, we rank preferred paths using a cost function, *cost_f (.),* which takes as arguments the vectors of path weight *w(f)* and constraint *L* vector. Following this ranking, the routing protocol advertises *k* paths of the lowest cost. Although different functions could be applied, the studies presented in [57],[59],[60],[61],[62] point out that the most effective are nonlinear, strict monotonic and convex functions. In our routing protocol, we use Minkowski norm of order *r*, defined as (24).

$$cost\_f(.) = \begin{cases} \left(\sum_{i=1}^{m} \left(\frac{w_i}{l_i} \right)^r \right)^{1/r} & , \quad w_i \le l_i \\ \infty & , \quad w_i > l_i \end{cases} \tag{24}$$

This function ranks preferred paths based on the distance of normalized weights $w_i^{(p)}$ from the point zero in *m*-dimensional decision space. Appropriate tuning of parameter *r* allows us to influence the shape of cost function.

For *r* equal to *1*, the path cost is linear combination of normalized path weights. Although this function can be easily interpreted, it is insensitive to unbalanced solutions, with extreme weights close to the constraint. Let us consider two exemplary feasible paths f^1 and f^2, with normalized weights $w(f^1)=[0.4,0.4]$ and $w(f^2)=[0.7, 0.1]$. The costs of both paths equal 0.8, while the probability of exceeding constraints is much higher for path f^2, because its first component is close to the constraint.

For $r \to \infty$, the cost is determined by the maximum component of the path's weight $w_i^{(f)}$, while the rest of the components are ignored. So, the cost of two exemplary paths f^3 and f^4, with normalized weights $w(f^3)=[0.8,0.1]$ and $w(f^4)=[0.8, 0.7]$ equals 0.8, while the probability of exceeding constraints is higher for path f^4.

In our algorithm, we assumed cost functions with *r* equal to 4, which is large enough to guarantee sensitiveness to unbalanced solution and it is still low enough to consider impact of all weights.

7.2 Simulation experiments

The simulation experiments aim to evaluate the performance of RAE and compare its effectiveness with the reference BGP-4 protocol. They are complementary to experiments carried out in the COMET federated testbed, reported in [9]. The experiments were performed in a multi-domain network that consists of 110, 512 and 830 domains. These networks were created based on the model presented in chapter 3, by aggregation of the domains proposed in [63]. Each domain uses RAE to calculate the inter-domain routing based on information about network topology and QoS provisioning of inter-domain links. Each inter-domain link is characterized by three QoS parameters related to COMET CoS: packet transfer delay, packet losses and bandwidth. The values of QoS parameters were chosen randomly from the range between 0 and 1. The experiments were repeated at least 100 times for different provisioning of inter-domain links.

The RAE performance was evaluated from the point of view of COMET CoS coverage. It is expressed by the average number of destination domains to which RAE could calculate at least one feasible path. The path is feasible if concatenated the value of link parameters on considered path

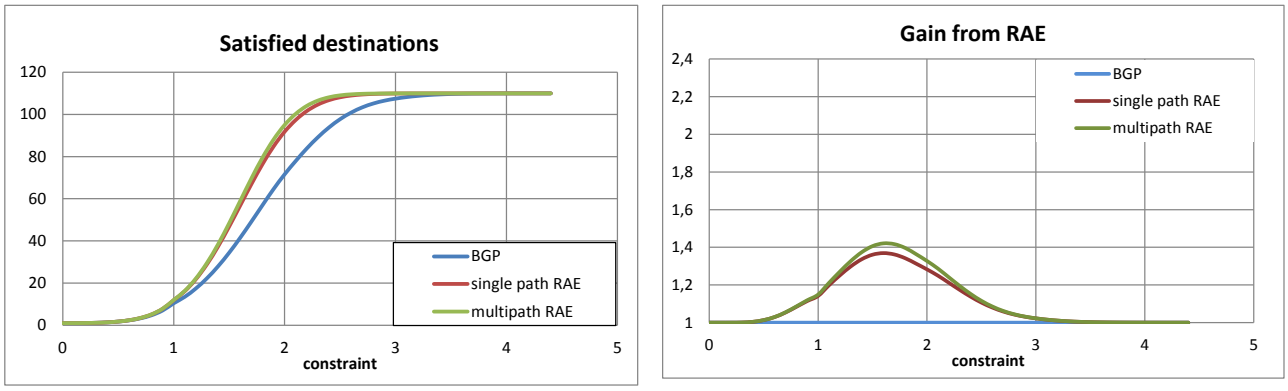
satisfies constraint assumed for all QoS parameters. This metric reflects RAE efficiency in providing CoS connectivity. In order to assess RAE results, we compare them with the results obtained for the shortest path routing performed by BGP-4 protocol.

In the simulation experiments, we measure the following metrics:

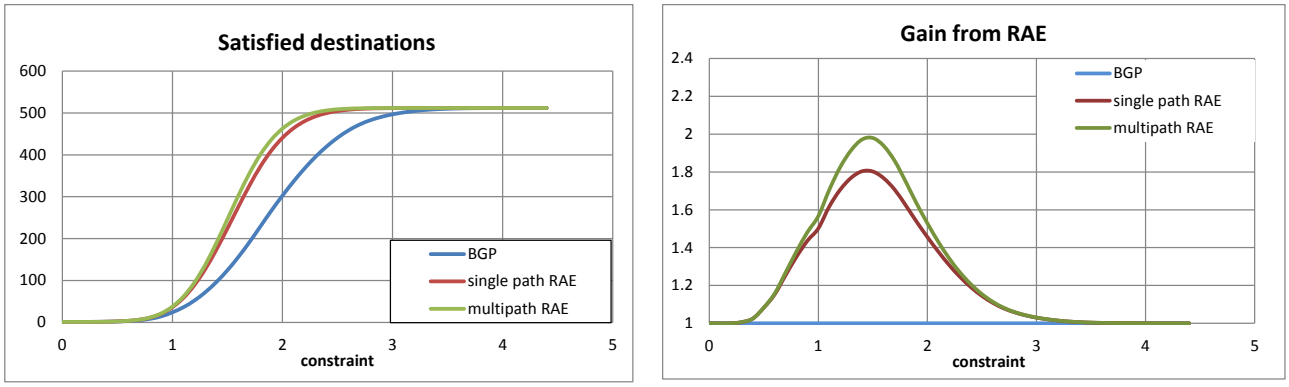
- **Number of satisfied destinations (NSD)** is the number of destination domains towards at least one feasible path exists (path which satisfies constraint)
- **Gain from RAE** is the ratio of the number of satisfied destination domains calculated by RAE to the number of satisfied destinations domains calculated by BGP4.

7.3 Results and conclusions

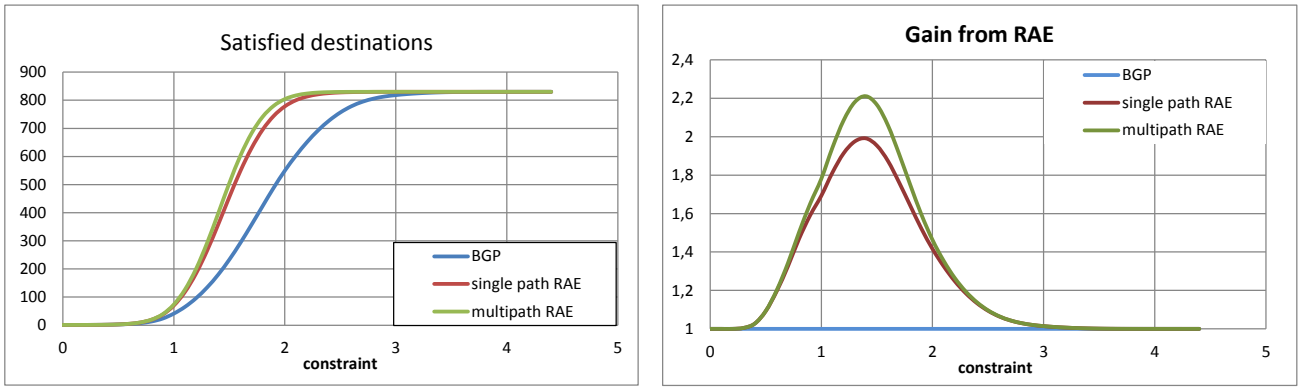
The number of satisfied destinations and gain from RAE was evaluated in multi-domain network. In order to assess the impact of the network size, we considered three networks consisting of 110, 512 and 830 domains, respectively.



(a) Multi-domain network with 110 domains



(a) Multi-domain network with 512 domains



(a) Multi-domain network with 830 domains

Figure 45: The performance of RAE for the network with 110, 512 and 830 domains.

Figure 45 presents results obtained for single path RAE, multi-path RAE and BGP-4. The RAE performance was analysed as a function of end-to-end QoS constrains, expressed by a factor of normalised values of all QoS parameters starting from 0, where no feasible paths exist, to 5 where all paths are feasible because the maximum path length is 5.

The presented results confirmed that RAE improves the coverage of COMET CoS in a multi-domain network by providing more content delivery paths satisfying constraints then the BGP4 protocol. The gain from RAE depends on the constraint values. For restrictive constraints, there are almost no feasible paths in the network so both RAE and BGP4 protocol provides similar results. On the other hand, in case of loose constrains almost all paths in the network are feasible so it does not mean what type of routing protocol is used. The maximum gain from RAE is visible for moderate constraints, which are in the middle of concatenated values related to the average path length.

The gain from RAE comes from its two new features: the multi-criteria path selection algorithm and the advertisement of multiple paths. However, the multipath feature only slightly improves the number of satisfied destinations (see results of multipath vs. single path RAE). The impact of multipath routing is more visible at the content consumption level studied in chapter 8.

Moreover, we can observe that the maximum gain from RAE increases in the network with the large number of domains. This effect comes from the fact that the number of alternative paths significantly increases with the number of domains. Consequently, the smart routing algorithm has more chance to find feasible paths.

8 Multi-criteria decision algorithm for ICN networks

In this chapter, we focus on performance evaluation of multi-criteria decision algorithm, which was developed in COMET to support multi-source and multi-path content resolution. The details of proposed algorithm are presented in [3] and was published in [13]. The enhanced version of the algorithm was published in joint COMET-ALICANTE paper [64]. The extended version of this paper is included in ANNEX C.

In order to ensure effective content delivery in multi-source environment, we maximize the information used in the server/path selection process by introducing the decision algorithm at two levels. The first level corresponds to routing awareness process in the network, which discovers content delivery paths between domains. The routing process is performed offline using long term information about network topology, COMET Classes of Service and the corresponding provisioned resources. The outcome of this process is a set of end-to-end content delivery paths between servers' and customers' domains, which are established to meet content transfer requirements of different types of content. The performance of routing awareness process is studied in chapter 7.

The second level of decision process selects the best server and path from the available candidates computed in the previous phase. This process is performed, independently in each consumer domain, upon receiving consumption request in an entity called decision maker. The decision maker is responsible for the selection of the best content server and path based on collected information about candidate servers and paths. In COMET system, the decision maker is located in the client CME.

In our algorithm, we consider: (1) the list of content servers that may stream the content and their load, and (2) the list of content delivery paths between particular server and consumer. Each path is characterised by path length, load on the path and QoS parameters. The complex set of parameters used for decision algorithm requires a multi-criteria decision algorithm. However, in contrast to the routing level process, there is no direct influence of one decision maker on another because decision spaces are independent. As a consequence, we may directly apply one of algorithms studied in the Multiple Criteria Decision Analysis (MCDA) [65],[66],[67]. We leverage the multi-criteria decision algorithm presented in [68], which uses some a priori knowledge about the problem in order to select the effective solution. Without loss of generality, our decision algorithm uses three decision variables related to server load, path length and bandwidth, although it could be easily extended to accommodate more decision variables. Our algorithm evaluates the impact of a particular decision variable using two reference parameters, called reservation level and aspiration level. The reservation level is the hard upper limit for the decision variable which should not be crossed by preferred solution. On the other hand, the aspiration level defines the lower bound for decision variable, beyond which preference of evaluated solutions is similar. The decision algorithm consists of three main steps:

Step 1: Decision Maker creates a decision space, which consists of the list of candidate solutions based on the information about servers and paths. Each candidate solution is a vector of decision variables, which has the following form:
Candidate solution [i]:

serverLoad – this is numerical representation of server status,
pathLength – the path length denotes the number of domains on the path. It is calculated based on the AS path parameter,
bandwidth – maximum supported bandwidth on the path.

Step 2: Decision maker calculates the rank value R_i for each candidate using objective function with reservation and aspiration levels specific for each decision variable.

$$R_i(.) = \min_{k=1,2,3}\left[\frac{r_k - q_k}{r_k - a_k}\right] \tag{25}$$

where: i is the number of candidate, k is the number of decision variable, q_k is current value of decision variable, r_k is a *reservation level* for decision variable k, while a_k is an *aspiration*

level for decision variable k, which is determined as a multiplication of reservation level by aspiration coefficient α_k, $a_k = \alpha_k \times r_k$.

Step 3: Decision Maker selects the candidate with maximum rank as the best solution. Note that considered aggregate objective functions may have more than one effective solution into the Pareto optimal set. Thus, some tie-breaking rules (e.g., lower server load is preferred or just random selection) are required to ensure only one solution is selected.

The operator of COMET may tune the behaviour of the decision algorithm by setting reservation and aspiration levels. In particular, he/she may reduce the importance of a given decision variable by setting the aspiration level close to the reservation or even completely exclude a given decision variable by setting the aspiration level equal to the reservation level. In this way, each domain may define its own decision strategy based on its own preferences. Below, we discuss exemplary decision strategies that can be enforced by our proposed algorithm.

Strategy 1: Random server. This strategy assumes that content server is selected randomly. It reflects the situation in the pure Internet without CDN support, where a user selects a content source without any information about available paths and servers.

Strategy 2: Closest server. This algorithm selects the server closest to the user. This approach reflects one of the strategies in the current CDNs [21], when information about server status is not available. In this case, decision algorithm should consider only *path length* (aspiration and reservation levels for *server load* and *bandwidth* are set to 1). In fact, in content network related literature, this simplistic approach is used in [42] and [69].

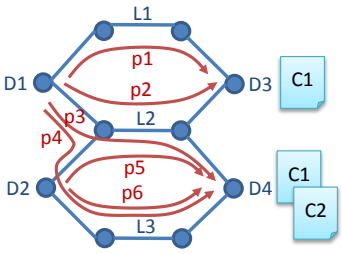
Strategy 3: The least loaded server (called best server strategy hereafter). This algorithm considers only the server load. It selects the least loaded server without considering information about paths. This strategy is used by most of P2P content delivery systems as well as some CDNs [21]. In this case, decision algorithm should consider only *server load* (aspiration and reservation levels for *path length* and *bandwidth* are set to 1).

Strategy 4: The best server and path. This algorithm considers both the *server load* and available bandwidth in the bottleneck link. It selects the least loaded server with the path of the best characteristics.

The above presented strategies will be evaluated in section 8.2. Note, that proposed decision algorithm also allows defining other decision strategies based on the subset of the available parameters.

8.1 Validation of decision algorithms in dynamic environment

The combination of both the decision algorithms at routing and at content consumption levels makes feasible, among others, load balancing in servers and network. Let us consider a simple scenario as presented in Figure 46(a). Consumers in domain D1 generate requests of the content C1, which can be found in both domain D3 and D4. The request arrival process is Poisson with mean rate = 1.0 req/s. At time t=500.0s, consumers in domain D2 start requesting (Poisson) for content C2 (located uniquely in a server in D4) with the same mean rate = 1.0 req/s. The parameters of the simulation are given in Figure 46 (b). The selected parameters are simplistic in order to understand the routing reaction of the algorithms to non-stationary congestion phenomena. Simulations that are more sophisticated are presented in section 8.2 for evaluating the performance in a large-scale model.



(a) simulation scenario

Consumers	D1	λ= 1 req/s	
	D2	λ= 1 req/s	
Content	C1	duration= 100 s	Streaming bandwidth= 100 kbps
	C2	duration= 100 s	Streaming bandwidth= 100 kbps
Network	all the links	Capacity= 100 Mbps	
Server	at D3	CPU Capacity= 100 parallel connections	
	at D4	CPU Capacity= 100 parallel connections	

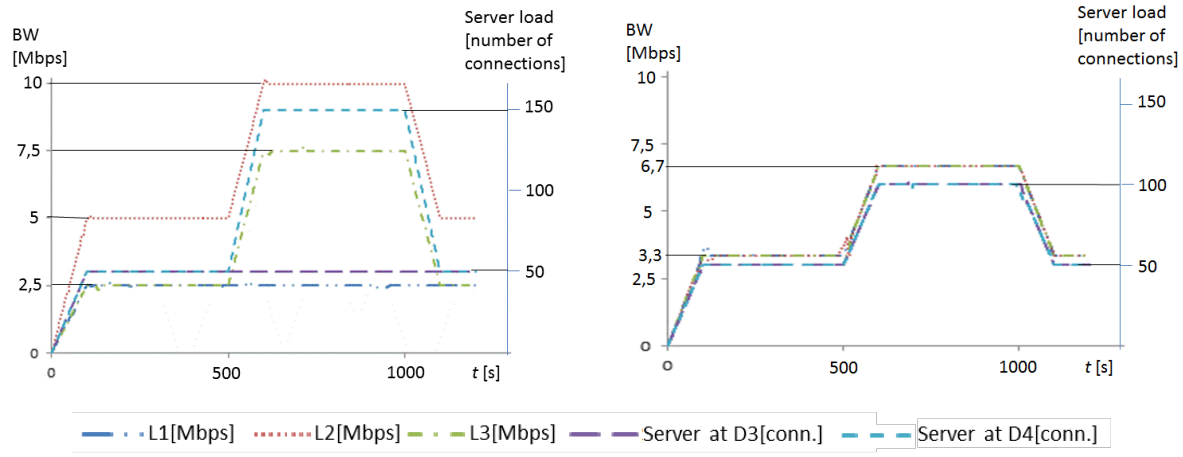
(b) simulation parameters

Figure 46: Scenario and parameters of the simulations for validation of decision algorithms.

We consider multi-path routing decision algorithm for 2 paths. So, the consumers in D1 can get the content C1 situated at D3 by: p1 or p2 paths, whereas the content C1 situated at D4 can be transferred by p3 or p4 paths, as indicated in Figure 46 (a). When the simulator allots a new request, the streaming server and path for serving the request are selected by considering both strategies: (1) random server and random path and (2) server load (reservation and aspiration levels are r_{server}=1.0 and a_{server}=0, respectively) and current available bandwidth in bottleneck link ($r_{bottleneck}$=100 Mbps; $a_{bottleneck}$=0 Mbps), where the bottleneck links are L1, L2 or L3, see Figure 46a. The server load is the relation between current served connections and CPU capacity of the server (100 connections served in parallel). The appropriate server is loaded by one more connection and the occupied bandwidth in the corresponding bottleneck link is increased by the streaming bandwidth of the content (100 kbps for both the contents), during the streaming duration of the content (100 s for both the contents).

Figure 47 shows the occupied bandwidth in the bottlenecks L1, L2 and L3 (left Y-axis of the figure) as well as the load of the two servers at D3 and D4 (right Y-axis) over the simulated period. Figure 47 (a) considers the random server and random path strategy. We observe that bottleneck L1 and server at D3 do not depend on the requested service in domain D2. So, the selected strategy does not adapt to the request process and no load balancing is achieved neither in bottlenecks nor in servers.

The second strategy, presented in Figure 47 (b), shows load balancing in network and servers thanks to our proposed decision algorithm. Note that the load in all links is the same regardless of content request calls. The same occurs for the server load in both the servers (in D3 and D4). When customers in D2 begin requesting content, the links L2 and L3 as well as server in D4 start over-loading. So, the new requests from D1 "prefer" the link L1 and the server in D3 achieving, in this way, balancing of bottlenecks and servers.



(a) Random path and random server strategy (b) Server load and available bandwidth strategy

Figure 47: Results of validation tests in dynamic scenario.

We can conclude that the decision algorithm positively reacts to the non-stationary congestion phenomena of the system, i.e., it may adapt to changing conditions.

8.2 Performance evaluation

The performance evaluation process runs as follows: within the video content consumption model described in chapter 3, consumers request for content following specific request arrival process (detailed in section 8.2.1). The Decision Maker is the responsible of selecting the server and path for serving the request depending on the evaluated strategy (detailed in section 8.2.3). The load of the selected server is increased by the requested bandwidth for whole duration of the content. In the same fashion, all the links of the selected path are loaded by a value equal to the bandwidth of the content during a time equal to the content duration. We assume that the connection bandwidth is independent of the state of the network during the content delivery, which is feasible for delivery of streaming content.

Whenever any link or server went beyond the capacity threshold, we consider that all the connections carried by the over-loaded link and/or all the connections currently served by the over-loaded server as unsuccessful (i.e., QoS was not guaranteed). Regardless of the state of the path and server, the request is served and the content is delivered by the selected path. When a connection is terminated, all relevant loads are taken off its respective link(s) and server. Finally, we measure the ratio of successful connections defined as the ratio of the number of successfully completed requests to the total number of content requests.

8.2.1 Request arrival process

The number of content request generated from each domain is proportional to the advertised prefixes of the domain. The arrival process for content requests is amply dealt in the literature; it depends on the type of content and type of application with most models suggesting a Poisson arrival process for short time scale (e.g. for IPTV applications in [70] and for Video on Demand applications in [71]). Some authors point out a modified Poisson process [19] for arrival rate. In our simulations, we applied the Poisson arrival model without considering non-stationary effects such as diurnal/night traffic. Each request is attached to a specific content following the Zipf's law for content popularity.

8.2.2 Routing protocols

The routing algorithms considered in the simulations are:

1. single shortest path,
2. multi shortest path,
3. single bandwidth-based path,
4. multi bandwidth-based path.

In all experiments, we use the RAE to calculate content delivery paths. However, we adapt its behaviour by setting the appropriate decision variable, i.e. path length or bandwidth, as well as by tuning the number of advertised paths, i.e. one or five paths. In order to evaluate how many preferred paths should be advertised, we assess the effectiveness of proposed protocol in the reference scenario where entire list of feasible paths was propagated. In this case, the protocol is more effective since there is no loss of information in the intermediate domains. Anyway, the results confirmed that effectiveness of routing protocol only slightly increases when protocol advertises more than five paths. Therefore, in our experiments we use five alternative paths.

Single shortest path protocol offers one shortest path between server and consumer domains. Therefore, in our routing model we randomly select one of the shortest paths. Multi shortest path protocol offers five the shortest paths between server and consumer domains and for each request one of them is selected following the rules of the decision algorithm. Single bandwidth-based path routing protocol offers the best path between server and client domains, where "best path" refers to

the path with highest capacity in the bottleneck link. Multi bandwidth-based path routing protocol offers five best paths between server and client domains.

8.2.3 Decision strategies

Let us recall analysed decision strategies:

1. random server and random path, which combined with shortest single path routing protocol, reflects the current Internet;
2. closest server and random path, which combined with shortest single path routing protocol, reflects the strategy of some current CDNs [72];
3. the least loaded server (called best server) and random path, where the least loaded server is the one that currently serves fewer requests;
4. the best server and the path with more available bandwidth in the bottleneck link (called best path).

Table 2 shows the values of reservation and aspiration levels for the parameters used in the simulations. The only strategy with two parameters is "best server / best path" where we believe that server load parameter is more crucial than bottleneck BW, i.e., between low loaded server and low loaded path, we select the former. Because of this, the reservation and aspiration level follows the formula (3), since the relation between aspiration and reservation level indicates the "a priori" importance of the parameter.

$$\frac{a_{server\_load}}{r_{server\_load}} < \left(\frac{a_{bottleneck\_BW}}{r_{bottleneck\_BW}}\right)^{-1} \quad (26)$$

Table 20: Reservation and aspiration levels in the simulations.

Strategy	Parameter	Reservation level (r)	Aspiration level (a)
Random server / random path	Not Applicable		
Closest server / random path	Path_length	r_1=100	a_1=0
Best server / random path	Server_load	r_1=1.0	a_1=0.0
Best server / best path	Server_load	r_1=1.0	a_1=0.0
	Bottleneck_BW	r_2=1500.0	a_2=1.5x10^5

In order to simplify the decision process when a new request arrives to the system, the group of the feasible solutions (Pareto optimal solution set) is reduced to 100 random servers (wherever they are) and one or five paths depending on the assumed routing.

8.2.4 Simulation results

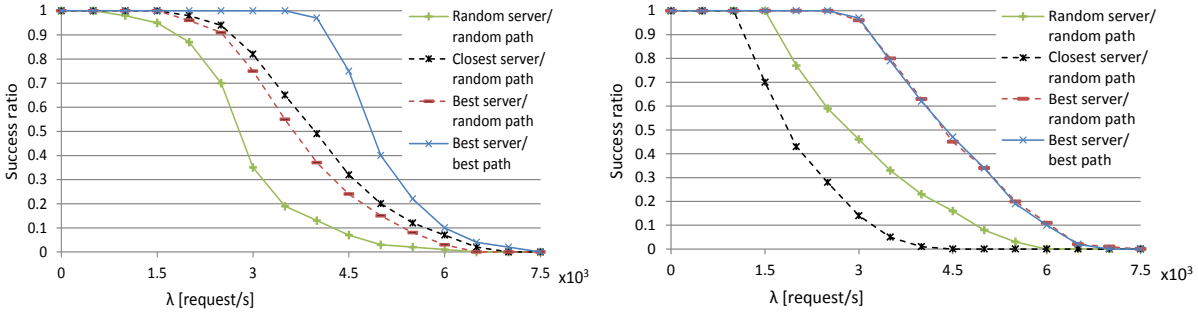
As aforementioned, a content delivery is considered as unsuccessful if one or both of the following are involved: over-loaded path or over-loaded server. Depending on the system state, one cause or the other becomes significant. In order to obtain the most complete results, we first investigate limit cases where both the causes appear simultaneously, and then we focus on deep analysis of working point.

8.2.4.1 Analysis of system under limit cases

For analysing the limit cases, we first performed tests for very high capacity threshold of the servers (10^8 concurrent connections) and capacity threshold in the link as assumed in the model of video content consumption. Figure 48(a) presents the success ratio (relation between successful

request and all attempts) for the four investigated decision strategies with single shortest path routing protocol. It can be observed that the request arrival rate for which the overload starts is in the range of 500 request/s.

In the following tests, we set to find the value of server capacity threshold for which the servers began to be overloaded in the range of 500 request/s. The link capacity threshold was infinite in all links in order to avoid overloaded links. Afterwards, we increased the server capacity threshold (the same value in all the servers) until we observed unsuccessful connections in the expected range, which occurred for a threshold of 200 concurrent connections. The results of these tests for single shortest path routing are presented in Figure 48(b).



(a) server threshold= 10^8 concurrent connections and link threshold= 1 Gbps

(b) server threshold= 200 concurrent connections and link threshold= 10^8Gbps

Figure 48: Success ratio for shortest single path routing algorithm and four decision algorithms.

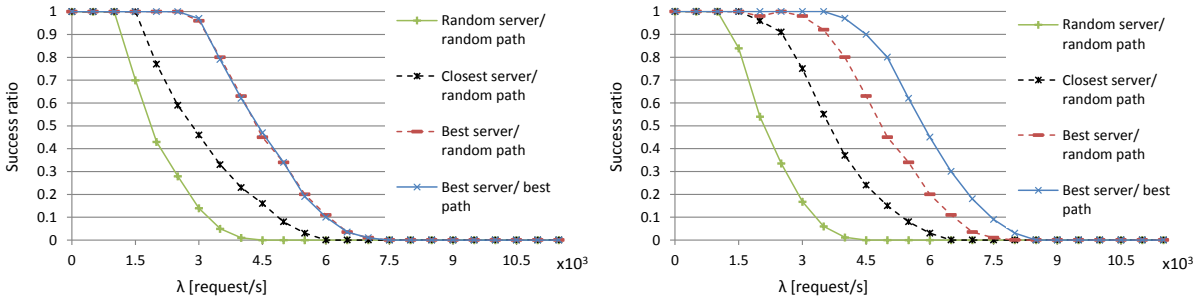
All the presented tests were repeated 5 times and all the results have confidence intervals fewer than 3% of the mean values at the 95% confidence level. For clarity purposes, we do not present the confidence intervals in the figures.

As we may observe in Figure 48(b), the two latter strategies (i.e., best server / random path and best server / best path) offered the same results since there were no bottleneck links in these simulations. In Figure 48(a), one could think that the strategies “closest server/ random path” and “best server/ random path” should offer the same results since the server is not the bottleneck in these simulations; nonetheless, “closest server” strategy makes sure that links are less used since the selected server is often in the consumer domain and no inter-domain link is loaded by the connection. Because of this, the results for this strategy are slightly better.

For other routing algorithms, the server load threshold value for which server and link overload appear simultaneously is also in the range of 200 concurrent connections. Therefore, in the next simulations, we set the threshold value of server load equal to 200 concurrent connections. Note that this value depends on the assumed scenario and thus, only valid for the presented results.

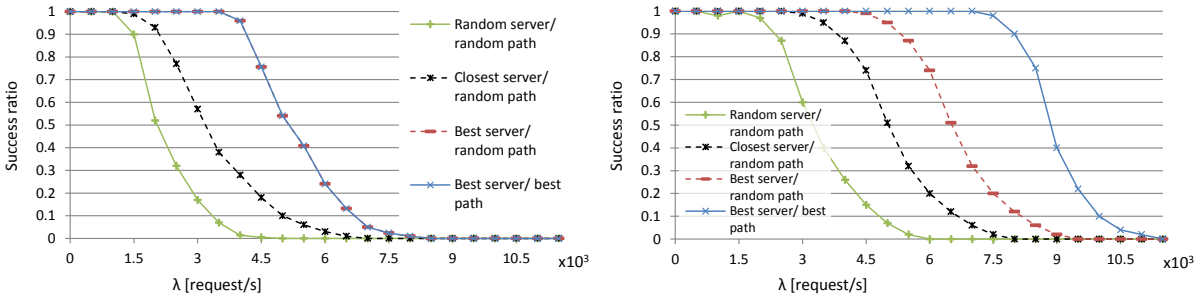
8.2.4.2 Analysis of system under working point

The next results compare the four decision algorithms and the four routing algorithms under working conditions defined from analysis of limit cases. Figure 49 presents the results for shortest (a) single and (b) multi path routing algorithms, whereas Figure 50 presents the results for best (a) single and (b) multi path routing algorithms.



(a) shortest SINGLE path routing algorithm (b) shortest MULTI path routing algorithm

Figure 49: Success ratio for the four decision algorithms and shortest path routing algorithm. Server threshold= 200 concurrent connections; link threshold= 1 Gbps



(a) best SINGLE path routing algorithm (b) best MULTI path routing algorithm

Figure 50: Success ratio for the four decision algorithms and best path routing algorithm. Server threshold= 200 concurrent connections; link threshold= 1 Gbps

As we may observe, the shape of the curves are different compared to those in Figure 48. The reason is the overlap of both overload effects: in servers and in links. The low confidence interval of the tests (fewer than 3% of the mean values at the 95% confidence level) validates the results.

The general conclusion of the results is that the increased knowledge about links and servers improves the efficiency of the system. Specifically, the combination of appropriate routing algorithm together with appropriate decision algorithm increases the effectiveness of the system. So, for example, for the assumed model and scenario, best multi path routing and best server / best path strategy ensure 90% of successfully delivered contents for arrival request rates up to 8.3×10^3 request/s, whereas in current Internet strategies (shortest single path routing and random server/ random path strategy) this same percentile is obtained only for arrival request rate fewer than 1.0×10^3 request/s.

We also observe that current CDNs strategies (single shortest path routing and closest server / random path strategy) reaches 90% success ratio for 2×10^3 request/s, whereas current Internet strategy reaches only 40% for the same arrival rate. Strategies which use more information about network (as best server/best path) obtain 100% successful content delivery for this case.

The strategies which use dynamic (on-line) information of the network and/or servers offer better results than static information, which suggests the need of monitoring systems in order to improve efficiency of the decision process. However, best single path results are not much better than shortest single path results because, in single path routing, the decision process cannot use this information since it cannot select the path. The efficiency for best multi path routing is much higher, which shows the importance of providing multi path routing protocol in networks delivering content.

By comparing results for the random server/random path strategy with shortest single path and shortest multi path routing algorithms, we conclude that, in the current Internet, multi path protocol does not introduce significant effectiveness gain.

In general, we can say that multi path routing protocol introduces the load balancing feature, which improves efficiency in all cases, not only in situations where overloaded link/server occurs. The same direction is followed by decision algorithms, such as best server/best path, which introduce repartition of the requests between servers and paths.

Let us remark that in best server/best path algorithm, we could tune reservation and aspiration levels obtaining a certain gain in efficiency, but this tuning depends on the assumed scenario and it is out of the scope of this paper.

8.2.5 Notes about the trustworthiness and reliability of the results

The complexity of the simulations prompts the question on the trustworthiness of the simulation process itself. We devoted much effort to understand the behaviour of the simulation process. After validating the correctness of our simulations in small scenarios, we concentrated on the possible reasons which could distort the results.

The high confidence of the results (lower than 3% of the mean values at the 95% confidence level) may be explained by the wide extension of the assumed model (video content Internet) as well as the fact that all the simulation tests counted at least 10^{11} served content requests. This should be sufficient to ensure dynamism in the states of servers and links (empty, loaded, overloaded) which are accounted for, i.e., both servers and links have changed state many times during each simulation test. In order to confirm this point, we performed tests for checking time range dependence.

For this, we analysed the behaviour of one randomly selected server and one randomly selected link. The number of served connections by the server through time as well as the capacity used in the link can be considered as stochastic processes, which are characterized by two dimensions: space and time. With regard to the space dimension, we assume that simulations behave as real content delivery in the network (for simplicity purposes) and we focus on verifying whether simulations and real networks behave similarly over time. In real networks, there appear many effects such as multiplexing with other traffic streams, which substantially reduce the time dimension dependence. By "time dimension dependence" we understand how much the state of given server, or link, in time t influences the state in time $t+\Box t$. Therefore, we should check short-range dependence also in our simulations. For this, we investigated and confirmed that the autocorrelation function for different lags of time in both server (number of served connections) and link (used capacity) decays slower than exponential function.

In conclusion, we verified that the simulation process does not enter in "loop stance" and the simulation process is trustworthy. However, the trustworthiness of the simulation process does not indicate that the results are reliable. The reliability of the results depends on the made assumptions, which are numerous in the presented simulations. Anyway, the comparison-based simulations credibly show the importance of network level information in increasing the efficiency of the systems. Our scope was checking whether more information ensures efficiency gain. By the term "ensure", we mean that in any case fewer information provides better results and on the other hand, more information provides significant improvement in efficiency. The results confirmed these two aspects.

8.3 Conclusions

In COMET, we proposed and evaluated the multi-criteria decision algorithm, which exploits two closely related processes. The first process, which operates offline in a long term, discovers multiple content delivery paths and gathers their respective transfer characteristics. The second process, which is invoked for each content request, combines available information about network and server condition for selecting the best content server and delivery path.

The simulation results confirmed that the two-level algorithm provides more information to the selection of server and path. This results in a higher percentile of satisfied content requests by improving utilization of network and server resources. When the number of content requests

increases, then the two-level algorithm makes feasible load balancing in both network and servers, avoiding or slowing down overload conditions. Load balancing is achieved also in situations of normal load, which might be an interesting feature for network and content service operators.

Further work will focus on the effect of the parameter setting in the efficiency gain in order to provide the best decision algorithm in content networks. Moreover, we started to analyse the optimization of multi-criteria decision algorithm by tuning the values of reference and aspiration levels for given set of parameters. In [64], the first results are presented. The enhanced studies are presented in draft paper included in Annex C. At last, when both the set of parameters and the tuning of reference levels are optimized, then it will be possible to study the difference between optimum and heuristic methods.

9 Summary and conclusions

In this deliverable, we have presented the analytical and simulation results corresponding to the scalability and performance evaluation of the COMET system, for both the decoupled and the coupled approaches.

For the case of the decoupled approach, the main results were obtained by simulation using a large-scale Internet model. More specifically, we focused on the effectiveness of the content resolution and content delivery processes, as key processes executed by the system in short time scale (on-line). The results confirmed that there is no critical element in the COMET system, which could lead to system performance degradation. More specifically, we can state that:

- The execution delays of the content resolution process in highly loaded COMET entities are satisfactory for the customers (the CRT delay is less that 2.5 s for 95 per cent of content requests);
- Even when the system size increases (e.g., content request rate, number of content sources, domain size), the CRT delays are still acceptable;
- The overhead of the COMET header, even in the worst case, is in similar range as IPv6 header;
- The CAFE performance is similar to the performance of an IP router.

Concluding, form the point of view of content resolution and delivery processes there are no barriers in deploying the COMET system in large-scale networks, like the Internet. On the other hand, we have pointed out that we can speed up the execution of the content resolution process by using caches and enhanced algorithms for the retrieval of server loads and paths.

From the point of view of the system scalability, it is important to note that in the decoupled approach, the control of the content resolution process is performed mainly by the client CME and the server CME, both of them located at the network edge domains (not in the transit domains). Hence, if we have problems with handing content requests in a given domain, we can offload the COMET entities by adding new instances in this domain. This is a very positive feature of the discussed system supporting its scalability. So, one can find some similarities with DiffServ architecture, which is regarded as a reference scalable solution. In this architecture the most important point is that the main processes are performed by edge routers while the core routers perform only forwarding. When traffic grows, we simply add new edge routers without changing the core.

The simulation studies of the coupled approach showed that the use of broadcast resolution does not offer any benefit over random-based resolution in terms of content delivery performance, making the latter more preferable. In addition, it was shown that the hop count does not vary considerably with topology size, making the coupled approach a purely feasible option from a scalability point-of-view, and hence resulting in worst-case content retrievals of well under one second.

As regards the in-network caching studies performed within the context of COMET, we conclude against the necessity of ubiquitous caching. Instead, we argue that more sophisticated algorithms can be applied to CAFEs, in order to optimise resource management and improve the performance of the new caching system. We have presented to approaches to in-network caching, both of which apply mainly to the coupled content resolution approach. The first one, the probabilistic caching approach, follows a resource management concept and optimises cache resources along a path of cache-equipped CAFEs. The second, the node centrality-based approach, finds the most popular paths and caches contents in those nodes. In doing this, the approach avoids caching redundancy and is shown to improve considerably the performance of the system. We therefore, conclude that in-network caching, in contrast to traditional overlay and hierarchical caching, can still provide performance benefit, but the related algorithms have to be designed in order to operate at line-speed. Our numerical results and conclusions (see end of chapter 6 for details) suggest that

performance benefit exceeds 10% compared to traditional approaches to content replication in cache systems.

The studies on multi-criteria decision algorithm confirmed that designed two-level algorithm provides more information to the selection of content server and path. This results in a higher percentage of satisfied content requests by improving utilization of network and server resources. When the number of content requests increases, the two-level algorithm makes feasible load balancing in both network and servers, avoiding or slowing down overload conditions. Load balancing is achieved also in situations of normal load, which is an interesting feature for network operators and content providers.

10 References

[1] COMET Deliverable, “D2.2: “High-Level Architecture of the COMET System”, January 2011

[2] COMET Deliverable, “D2.3: Global Architecture of the COMET System”, February 2013

[3] COMET Deliverable, “D3.2: Final Specification of Mechanisms, Protocols and Algorithms for the Content Mediation System”, November 2011.

[4] COMET Deliverable, “D4.2: Final Specification of Mechanisms, Protocols and Algorithms for Enhanced Network Platforms”, December 14th, 2011.

[5] COMET Deliverable, “D3.3: Prototype Implementation and System Integration Interfaces for the Content Mediation System” January 2012

[6] COMET Deliverable, “D4.3: Prototype Implementation and System Integration Interfaces for Enhanced Network Platforms”, May 2012.

[7] COMET Deliverable, “D5.1: Integration of COMET Prototype and Adaptation of Applications”, July 2012.

[8] COMET Deliverable, “D6.1: Demonstration Scenarios and Test Plan”, June 2012.

[9] COMET Deliverable, “D6.2: Prototype Experimentation and Demonstration”, February 2013.

[10] G. Garcia, A. Beben, F. J. Ramon, A. Maeso, I. Psaras, G. Pavlou, N. Wang, J. Sliwinski, S. Spirou, S. Soursos, E. Hadjioannou "COMET: Content Mediator Architecture for Content-aware Networks", in Future Network and Mobile Summit 2011, Warsaw, Poland, 15-17 June 2011.

[11] G. Pavlou, N. Wang, W. K. Chai and I. Psaras, "Internet-scale Content Mediation in Information-centric Networks", Annals of Telecommunications, Special Issue on Networked Digital Media, Springer 2012, (DOI) 10.1007/s12243-012-0333-8.

[12] W. K. Chai, et al., “CURLING: Content-ubiquitous resolution and delivery infrastructure for next-generation services,” IEEE Communications Magazine, vol. 49, no. 3, 2011, pp. 112-120.

[13] A. B□ben, J. Mongay Batalla, W. K. Chai and J. □liwi□ski, "Multi-criteria Decision Algorithms for Efficient Content Delivery in Content Networks", Annals of Telecommunications, Special Issue on Networked Digital Media, Springer 2012, (DOI) 10.1007/s12243-012-0321-z.

[14] CAIDA dataset; http://www.caida.org/research/topology/#Datasets

[15] CAIDA, The CAIDA AS Relationships Dataset, 2010. Available from: <http://www.caida.org/data/active/as-relationships/>.

[16] University of Oregon Route Views Archive Project David Meyer, http://archive.routeviews.org/.

[17] RIPE Network Coordination Centre, “Routing Information Service”, http://www.ripe.net/data-tools/stats/ris/ris-peering-policy.

[18] R.V. Oliveira, D. Pei, W. Willinger, B. Zhang, L. Zhang, “In search of the elusive ground truth: the Internet’s AS-level connectivity structure”, SIGMETRICS Perf. Eval. Rev. 36 (2008) 217–228.

[19] H. Yu, D. Zheng, B. Zhao and W. Zheng, “Understanding User Behavior in Large-Scale Video-on-Demand Systems”. In Proc of EuroSys 2006.

[20] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian and M. Karir, “ATLAS Internet Observatory 2009 Annual Report”. Arbor Networks Inc., University of Michigan and Merit Network Inc. 2009.

[21] R. Buyya, M. Pathan and A. Vakali, “Content Delivery Networks”, ISBN 978-3-540-77886-8, Springer-Verlag, Germany, 2008.

[22] M. Zink, K. Suh, Y. Gu, and J. Kurose, “Watch global, cache local: YouTube network traces at a campus network - measurements and implications”,' Proc. IEEE Multimedia Computing and Networking, 2008.

[23] Akamai Technologies, Inc., “Facts & Figures”, http://www.akamai.com/html/about/facts_figures.html.

[24] H. Bidgoli, The Internet Encyclopedia (Editor-in-Chief), John Wiley & Sons, Inc., Hoboken, NJ, 2004. (3-volumes). ISBN: 0471222046. Volume 2, page 502.

[25] A. Nimkar, C. Mandal and C. Reade, “Video Placement and Disk Load Balancing Algorithm for VoD Proxy Server”, In the proceedings of 3rd IEEE international conference on Internet Multimedia Services Architecture and Applications, 2009, pp. 141-146,.

[26] Hitachi, Ltd., “Hitachi VOD Server”. ©2010, http://www.hitachi.com/products/it/network/SDP/.

[27] NetUP Inc., “IPTV solutions by NetUP: Video on Demand & Virtual Cinema”, ©2011, http://www.netup.tv/en-EN/vod-nvod-server.php.

[28] VBrick Systems Inc., “VBrick Enterprise Media System”. ©2010, http://www.vbrick.com/docs/vbrick_datasheet_VOD-W-family.pdf.

[29] J. Donovan and N. Faris, Akamai Technologies, Inc., “Digital Movie Traffic on the Akamai Network Increases Three Fold”, March 2010, http://www.akamai.com/html/about/press/releases/2010/press_031610_1.html.

[30] Film Web Inc., homepage: http://www.filmweb.pl/.

[31] K. Florance, Netflix Content Delivery, “Netflix Performance on Top ISP Networks”. January 2011, http://techblog.netflix.com/2011/01/netflix-performance-on-top-isp-networks.html.

[32] K. Gummadi et al., “Measurement modeling and analysis of a peer-to-peer file sharing workload”. Proc. of SOSP. October 2003.

[33] J. Kangasharju, J. Roberts, and K. W. Ross, “Object Replication Strategies in Content Distribution Networks”, Computer Communications, 25(4), Apr. 2002, pp. 367-383.

[34] Leonard Kleinrock, Queueing Systems: Volume II - Computer Applications, Wiley Interscience, New York, 1976

[35] D.Gross, C. Harris, Fundamentals of queuing theory (4th ed.), Wiley Interscience, New Jersy, 2008

[36] Darren R. Law: “Scalable means more than more: A Unifying definition of simulation scalability”. Proceedings of the 1998 Winter Simulation Conference. 1998

[37] Cheng Huang, Angela Wang, Jin Li and Keith W. Ross, “Measuring and evaluating large-scale CDNs”, In Proceedings of the 8th ACM SIGCOMM conference on Internet measurement (IMC '08),. ACM, New York, NY, USA, 2008, pages 15-29

[38] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss. IETF RFC 2475: “An Architecture for Differentiated Service”. December 1998

[39] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine. IETF RFC 2998: “A Framework for Integrated Services Operation over Diffserv Networks”. November 2000

[40] B. Zhang, T. Ng, A. Nandi, R. Riedi, P. Druschel, and G. Wang, “Measurement-based analysis, modelling, and synthesis of the Internet delay space”, in Proc. ACM SIGCOMM IMC’06, 2006, pp. 85-98.

[41] H. Tangmunarunkit, J. Doyle, R. Govindan, W. Willinger, S. Jamin, and S. Shenker, “Does AS size determine degree in AS topology”, ACM SIGCOMM Computer Communications Review 36, 2001, pp. 7-8.

[42] V. Jacobson, et. al., “Networking Named Content,” Proc. ACM CoNEXT, 2009, pp.1-12.

[43] I. Psaras, W. K. Chai and G. Pavlou, “Probabilistic In-Network Caching for Information-Centric Networks”, ACM SIGCOMM ICN Workshop 2012.

[44] N. Laoutaris, H. Che and I. Stavrakakis, “The LCD interconnection of LRU caches and its analysis,” Performance Evaluation, vol. 63, no. 7, pp. 609-634, Jul. 2006.

[45] W. K. Chai, D. He, I. Psaras and G. Pavlou, “Cache 'Less for More' In Information-Centric Networks”. IFIP NETWORKING 2012, Prague, Czech Republic, May 2012

[46] W. K. Chai, D. He, I. Psaras and G. Pavlou "Cache ‘Less for More’ in Information-Centric Networks (Extended Version)", Elsevier Computer Communications, Special Issue on Information-Centric Networking, 2013, (DOI) 10.1016/j.comcom.2013.01.007.

[47] A. L. Barabasi and R. Albert, “Emergence of scaling in random networks,” Science, vol. 286, no. 5439, pp. 509-512, Oct. 1999.

[48] M. Everett, S. Borgatti, “Ego network betweenness,” Social Networks, 27(2005) pp. 31-38.

[49] P. Pantazopoulos, M. Karaliopoulos and I. Stavrakakis, “Centrality-driven scalable service migration,” Proc. International Teletraffic Congress (ITC), 2011.

[50] A. Ghodsi, et. al., “Information-centric Networking: Seeing the forest for the trees,” ACM Workshop on Hot Topics in Networks (HotNets-X), Cambridge, MA, Nov. 2011.

[51] H. Che, Y. Tung and Z. Wang, “Hierarchical web caching systems: modelling, design and experimental results,” IEEE Journ. on Selected Areas of Communications, 20(7), 2002.

[52] T. M. Wong, J. Wilkes, “My cache or yours? Making storage more exclusive,” Proc. USENIX Annual Technical Conference, Monterey, CA, 2002, pp. 161-175.

[53] S. Wassermann and K. Faust, “Social Network Analysis: Methods and Applications”,' Cambridge: Cambridge University Press, 1994.

[54] K.-I. Goh, B. Kahng and D. Kim, “Universal Behavior of Load Distribution in Scale-Free Networks”,' Physical Review Letters, vol. 87, no. 27, 31 Dec. 2001.

[55] K.-I. Goh, et. al., “Load distribution in weighted complex networks”,' Physical Review E 72, 017102 (2005)

[56] H. Wang, J. M. Hernandez and P. Van Mieghem, “Betweenness centrality in a weighted network”,' Physical Review E 77, 046105, 2008.

[57] X. Masip-Bruin et al., “Research challenges in QoS routing”, Computer Communications, vol. 29, no. 5, March 2006, pp. 563-581.

[58] F. Kuipers et al., “Performance evaluation of constraint-based path selection algorithms," IEEE Network, vol.18, no.5, September-October. 2004, pp. 16- 23

[59] G. Cheng, N. Ansari, “On selecting the cost function for source routing”, Computer Communications, vol. 29, issue 17, 2006, Elsavier, pp.3602-3608.

[60] G. Cheng, “The revisit of QoS routing based on non-linear Lagrange relaxation”, Int. J. Commun. Syst., vol. 20, 2007

[61] P. Khadivi, S. Samavi, and T. D. Todd, “Multi-constraint QoS routing using a new single mixed metrics”, *J. Netw. Comput. Appl., vol.* 31, no. 4, November 2008, pp. 656-676.

[62] P. Mieghem, F.A. Kuipers, “Concepts of exact QoS routing algorithms”, IEEE/ACM Trans. Netw., vol. 12, no. 5, October 2004, pp.851-864.

[63] B. Premore, “Multi-AS Topologies from BGP Routing Tables”, http://www.ssfnet.org/Exchange/gallery/asgraph/index.html, 2005

[64] J. Mongay Batalla, A. B□ben, and Y. Chen and, “Optimization of decision process in Network and Server-aware algorithms”. 978-1-4673-1391-9/12 ©2012 IEEE Networks 2012, Rome (Italy). October 2012.

[65] M. Ehrgott, “Multicriteria Optimization”. Springer-Verlag New York, Inc. 2005. ISBN: 3540213988

[66] A.P. Wierzbicki, M. Makowski, J. Wessels, “Model-based decision support methodology with environmental applications”, Kluwer Academic Publishers. Dordrecht, NL. 2000. ISBN: 0-7923-6327-2

[67] Y. Sawaragi; H. Nakayama and T. Tanino. “Theory of Multiobjective Optimization”. Mathematics in Science and Engineering, vol. 176 Academic Press Inc. ISBN 0126203709. 1985.

[68] A. Wierzbicki, “The use of reference objectives in multiobjective optimization”. Lecture Notes in Economics and Mathematical Systems, vol. 177. Springer-Verlag, pp. 468–486

[69] 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 Proc. ACM SIGCOMM ’07, Kyoto, Japan, Aug. 2007.

[70] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, “Watching television over an IP network”. Proc. of the 8th ACM SIGCOMM Conference on Internet Measurement, Vouliagmeni, Greece, October 20 - 22, 2008.

[71] H. Yu, D. Zheng, B. Zhao, and W. Zheng, “Understanding user behavior in large-scale video-on-demand systems”. SIGOPS Oper. Syst. Rev. 40, 4 (Oct. 2006), pp. 333-344.

[72] M. Hofmann, R. Leland, R. Beaumont, “Content Networking: Architecture, Protocols, and Practice”, Morgan Kaufmann Publisher. ISBN 1-55860-834-6. 2005.

11 Abbreviations

A-Cache	Authoritative Cache
AS	Autonomous System
B-A	Barabasi-Albert
Betw	Betweenness
BGP	Border Gateway Protocol
BW	Bandwidth
CACF	Cache-Aware Caching Function
CAFE	Content Aware Forwarding Entity
CAIDA	Cooperative Association for Internet Data Analysis
CC	Content Client
CCN	Content Centric Network
CDN	Content Delivery Network
CFP	Content Forwarding Plane
CME	Content Mediation Entity
CMF	Content Mediation Function
CMFI	Content Multiplexing Fairness Index
CMP	Content Mediation Plane
CoI	Content of Interest
CoS	Class of Service
CP	Content Publisher
CR	Content Record
CR-Cache	Content Record Cache
CRE	Content Resolution Entity
CRL	Content Retrieval Latency
CRME	Content Resolution and Mediation Entity
CRSR	Content Retrieval Success Ratio
CRT	Content Resolution Time
CS	Content Server
DiffServ	Differentiated Service s
DNS	Domain Name System
FIFO	First In First Out
ICN	Information Centric Networks
IP	Internet Protocol
IPLR	IP Packet Loss Ratio
IPTD	IP Packet Transfer Delay
IPv6	IP version 6

ISP	Internet Service Provider
LCD	Leave Copy Down
LCEd	Leave Copy Edge
LRU	Least Recently Used
MCDA	Multiple Criteria Decision Analysis
MPLS	Multiprotocol Label Switching
NLRI	Network Layer Reachability Information
NR	Name Resolution sub-process
NSD	Number of satisfied destinations
PC	Path Configuration sub-process
P-Cache	Path Cache
Pdf	Probability density function
PLR	Path and Load Retrieval sub-process
ProbCache	Probabilistic In-Network Caching
QoS	Quality of Service
RAE	Routing Awareness Entity
RCT	Routing Convergence Time
Rdm	Random caching strategy
RTT	Round Trip Time
SL-Cache	Server Load Cache
SMA	Server Monitoring Agent
SNME	Server and Network Monitoring Entity
SQA	Smart Querying Algorithm
TSB	Time Since Birth
URL	Universal Resource Locator
VoD	Video On Demand

12 Annex A: Paper submitted to IEEE TPDS

In Annex A, we have included draft paper about probabilistic in-network caching, discussed in section 6.2. This draft was submitted to IEEE Transaction on Parallel and Distributed System on November 2012.

IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. X, NO. X, NOVEMBER 2012 1

In-Network Cache Management and Resource Allocation for Information-Centric Networks

Ioannis Psaras, *Member, IEEE*, Wei Koong Chai, *Member, IEEE* and George Pavlou, *Senior Member, IEEE*

Abstract

We introduce the concept of *resource management* for in-network caching environments. We argue that in *Information-Centric Networking* environments, deterministically caching content messages at predefined places along the content delivery path results in unfair and inefficient *content multiplexing* between different content flows. Instead, introducing sophisticated mechanisms to allocate resources along the path according to content flow characteristics results in better use of network resources and therefore, higher overall performance.

We depart from the observation that caching *every* content message at *every* router that this message traverses causes huge *caching redundancy* in Information-Centric Networks. The design principles of our proposed in-network caching scheme, which we call *ProbCache*, target these two outcomes, namely *reduction of caching redundancy* and *fair content flow multiplexing* along the delivery path. In particular, *ProbCache* approximates the *caching capability* of a path and caches contents probabilistically to: *i)* leave caching space for other flows sharing (part of) the same path, and *ii)* fairly multiplex contents in caches along the path from the server to the client.

We evaluate the proposed schemes in both homogeneous and heterogeneous cache size environments and formulate a framework for resource allocation in in-network caching environments. The proposed probabilistic approach to in-network caching exhibits ideal performance both in terms of *network resource utilisation* and in terms of *resource allocation fairness* among competing content flows. Finally, and in contrast to the expected behaviour, we find that the efficient design of *ProbCache* results in fast convergence to caching of popular content items.

Index Terms

Information-Centric Networks, In-Network Caching, Content Multiplexing, Cache Capacity,

The authors are with University College London, Dept. of Electrical and Electronic Engineering. Contact addresses: Ioannis Psaras (i.psaras@ucl.ac.uk), Wei Koong Chai (w.chai@ucl.ac.uk), George Pavlou (g.pavlou@ucl.ac.uk).

November 1, 2012 DRAFT

13 Annex B: Paper submitted to Computer Networks

In annex B, we have included draft paper about in-network caching that was submitted to Elsavier Computer Networks on January 2013.

In-Cache Time Estimations for Information-Centric Networks

Ioannis Psaras, Richard G. Clegg, Raul Landa, Wei K. Chai, George Pavlou

Dept. of Electronic and Electrical Engineering
University College London
Torrington Place WC1E 6EA
London, UK

Abstract

Information-Centric Networks (ICNs) give the opportunity to cache named content in in-network, on-path, or off-path caches. In this paper, we provide a model that calculates the proportion of time that a specific content, called the *Content of Interest*, or *CoI*, stays in a network cache. Our model takes as input the rate of requests for the *CoI* at a given cache (denoted as λ) and the approximate rate of requests for all other contents that this cache is serving (denoted as μ). Effectively, the sum of these two rates (*i.e.*, $\lambda + \mu$) largely represents the amount of traffic that a cache is serving per unit time.

We begin from a single cache and model the journey of the *CoI* from the first position in the cache to its eviction using continuous time Markov chains, assuming Least Recently Used (LRU) caches. We then extend our model for the case of multiple content caches along a delivery path, assuming Poisson arrivals of requests. We verify the validity of our model using extensive simulation tests; we show that indeed, our model predicts the proportion of time that the *CoI* stays in the cache with 90% precision when the exponent of the Zipf distribution is relatively small (*i.e.*, for values smaller than 1.3).

We believe that our model will prove to be a very useful tool for network operators as it provides the opportunity to approximate the hit ratio of specific contents, given their relative request rate. In turn, the approximation of hit/miss rates can drive the design, setup and management of a network of caches, as the ICN paradigm proceeds to deployment. Examples of the applicability of our model include sizing in-network caches by taking into account business/billing models between ISPs, CDNs and content providers.

Keywords:
Information-Centric Networks, In-Network Caching, Modelling Cache Networks, Time In Cache Estimation

Email addresses: i.psaras@ucl.ac.uk (Ioannis Psaras), r.clegg@ucl.ac.uk (Richard G. Clegg), r.landa@ucl.ac.uk (Raul Landa), w.chai@ucl.ac.uk (Wei K. Chai), g.pavlou@ucl.ac.uk (George Pavlou)

Preprint submitted to Computer Networks *January 9, 2013*

14 Annex C: Paper submitted to Springer JTS

In Annex C, we included the joint COMET-ALICANTE paper about extended decision algorithm for ICN networks that has been submitted to Springer Journal of "Telecommunication Systems" on January 2013. After publication on IEEE Networks 2012 conference [64], we have been invited to submit enhanced version to this journal.

Draft version

Optimized decision algorithm for Information Centric Networks

Jordi Mongay Batalla[1], Andrzej Bęben[2] and Yiping Chen[3]

[1]*National Institute of Telecommunication*
Warsaw, Poland
+48 609 709 150
jordim@interfree.it
http://www.nit.eu/z3
[2]*Warsaw University of Technology*
Warsaw, Poland
abeben@tele.pw.edu.pl
http://tnt.tele.pw.edu.pl
[3]*CNRS / LaBRI*
University of Bordeaux
Talence, France
yiping.chen@labri.fr

Information Centric Networks enable network, server context and user context-awareness, to achieve an enhanced architecture for the delivery of the multimedia content. The information comes from different sources and serves as input for the decision algorithms for choosing the pertinent configuration such as the best server or the suitable delivery path. Therefore, the relevance of the input information and the efficiency of the decision algorithms are both crucial for the system performance.

This paper discusses how the multi-criteria optimization algorithms could be applied in the context of the ICN. Based on the approach of the Reference Level decision, an optimized algorithm is proposed to take into account the impact of different network and server parameters, and dynamically adapt the decision to the current state of the system. The added value of this paper is the extensive simulation model based on real Video on Demand services that allows comparing different decision algorithms in Information Centric Networks. The simulation results prove the efficiency of the proposal and suggest its deployment on the future media networks.

Decision process, Content Aware Network, Network Awareness, Multi-criteria algorithm, Reference Level algorithm, Information Centric Network

1