



# **Content Mediator architecture for content-aware nETworks**

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

## **Deliverable D3.2 Final Specification of Mechanisms, Protocols and Algorithms for the Content Mediation 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

**© Copyright 2012, the Members of the COMET Consortium**

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

Prof. George Pavlou  
University College London, g.pavlou@ee.ucl.ac.uk  
Department of Electronic & Electrical Engineering  
University College London  
Torrington Place, London,  
WC1E 7JE

Phone: +44 (0)20 7679 3985  
E-mail: comet@comet-project.org

## Document Control

**Title:** Final Specification of Mechanisms, Protocols and Algorithms for the Content Mediation System

**Type:** Public

**Editor(s):** Wei Koong Chai, Ioannis Psaras

**E-mail:** wchai@ee.ucl.ac.uk, i.psaras@ee.ucl.ac.uk

**Author(s):** Wei Koong Chai (UCL), Ioannis Psaras (UCL), Marinos Charalambides (UCL), Daphne Tuncer (UCL), George Pavlou (UCL), David Flórez (TID), Francisco Javier Hernández Romero (TID), Andrzej Beben (WUT), Jordi Mongay Batalla (WUT), Jarek Sliwinski (WUT), Piotr Wisniewski (WUT), Wojciech Burakowski (WUT), Ning Wang (UniS), George Kamel (UniS), Spiros Spirou (ICOM), Michael Georgiades (PrimeTel)

**Doc ID:** D3.2-v2.1.doc

## AMENDMENT HISTORY

Version	Date	Author	Description/Comments
Vo.1	July 29, 2011	Wei Koong Chai	Release of draft ToC
Vo.2	September 23, 2011	Florez, David	TOC for decoupled
v.03	October 10 <sup>th</sup> , 2011	David Flórez, Francisco Javier Hernández Romero	TID's initial contribution
v.04.1			
v.04.2	October 13 <sup>th</sup> , 2011	Andrzej Beben, Jordi Mongay Batalla, Jarek Sliwinski	WUT's contributions about path discovery, decision process, path configuration and evaluation of decision strategies
V.04.3	October 14 <sup>th</sup> , 2011	David Flórez	Some cleaning up the decoupled Approach
Vo.5	October 17 <sup>th</sup> , 2011	Wei Koong Chai	First release of a partial draft (integrated contribution from both coupled and decoupled approaches)
Vo.5.2	October 20 <sup>th</sup> , 2011	David Flórez	Acronyms list updated. Further cleaning up for the Decoupled
Vo.5.3	October 25 <sup>th</sup> , 2011	Wei Koong Chai	Release with executive summary and introduction
V.0.5.4	November 10 <sup>th</sup> 2011	Wei Koong Chai	Release of the complete full draft
V.0.5.5	November 14 <sup>th</sup> 2011	Wei Koong Chai	Final version
V1.0	December 5 <sup>th</sup> , 2011	David Flórez	TID's contribution to amendment
V1.1	December 14 <sup>th</sup> , 2011	Wei Koong Chai	First release of the full amended version.
V1.2	December 19 <sup>th</sup> , 2011	David Flórez	Changes in decoupled terminology
V2.0	December 20 <sup>th</sup> 2011	David Flórez	Official Final version of the document
V2.1	June 1 <sup>st</sup> 2012	David Flórez	Copyright corrected

### Legal Notices


The information in this document is subject to change without notice.

The Members of the COMET Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the COMET Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

## Table of Contents

<b>Prof. George Pavlou</b>	<b>1</b>
<b>1 Executive Summary</b>	<b>7</b>
<b>2 Introduction</b>	<b>8</b>
<b>3 Decoupled Approach Specifications</b>	<b>10</b>
3.1 Overview	10
3.2 Entities and Interactions	10
3.3 Content Publication Process	19
3.4 Content Resolution Process	23
3.4.1 Content Request	23
3.4.2 Content Resolution	24
3.4.3 Path Discovery	25
3.4.4 Server Awareness	28
3.4.5 Decision Process	33
3.4.6 Path Configuration	36
3.4.7 CAFE Configuration	39
<b>4 Coupled Approach Specifications</b>	<b>41</b>
4.1 Overview	41
4.2 Basics	42
4.2.1 Entities	42
4.2.2 Interactions	43
4.2.3 Provider Route Forwarding Rule	44
4.3 Content Publication	45
4.4 Content Resolution	48
4.5 Preparation for Content Delivery	51
4.6 Discussions	52
4.6.1 Scalability	52
4.6.2 Security	53
<b>5 Server-awareness in COMET system</b>	<b>54</b>
5.1 Introduction	54
[REDACTED]	
[REDACTED]	
[REDACTED]	
[REDACTED]	
5.3 Server-awareness in the Coupled Approach	62

---

5.3.1	<i>Design Overview</i>	62
5.3.2	<i>Dissemination of Server information</i>	62
5.3.3	<i>Server-load Aware Resolution</i>	65
5.3.4	<i>Discussion</i>	66
<b>6</b>	<b>Evaluation of decision strategies for CAN</b>	<b>68</b>
6.1	Simulation model	68
6.2	Considered decision strategies	68
6.3	Simulation results	69
6.3.1	<i>Comparison of decision strategies</i>	69
6.3.2	<i>Impact of admission control</i>	71
6.4	Summary	72
<b>7</b>	<b>Summary and Conclusions</b>	<b>73</b>
<b>Annex A</b>	<b>Model for Content Internet</b>	<b>75</b>
<b>Annex B</b>	<b>Decoupled Specification Details</b>	<b>77</b>
B.1	Content Request Specification Details	77
B.1.1	<i>Content Request, Query Datagram Structure</i>	77
B.1.2	<i>Content Request, Response Datagram Structure</i>	79
B.2	Path Configuration Specification Details	82
B.2.1	<i>Information required for path configuration</i>	82
B.2.2	<i>Path configuration process</i>	85
<b>Annex C</b>	<b>Server Awareness Specification Details</b>	<b>91</b>
C.1	Decoupled Approach	91
C.1.1		
<b>8</b>	<b>References</b>	<b>93</b>
<b>9</b>	<b>Abbreviations</b>	<b>96</b>
<b>10</b>	<b>Acknowledgements</b>	<b>98</b>

(This page is left blank intentionally.)

# 1 Executive Summary

In this deliverable, we present the final specification of our Coupled and Decoupled content publication and resolution algorithms for multi-domain systems. These approaches were initially discussed in D3.1 and belong to the Content Mediation Plane (CMP) of the COMET architecture.

In particular, Section 2 introduces the technical issues discussed in this document. Section 3 details the Decoupled approach. In this section, we present details of the involved high-level entities as well as the interactions between them. The detailed process of the Content Publication and Content Resolution are provided. This includes how a content is published and how a content request is handled following this approach. Related content resolution processes such as path discovery, selection and configuration are also specified.

In Section 4, we give the specifications for the Coupled approach. Similar to the previous section, we first discuss the specifications of the involved entities, together with their functions and interactions. details of the *Content Publication* and *Content Resolution* processes following the *provider route forwarding rule* is explained. Due to the tightly coupled nature of the approach on content resolution and delivery, this section also specifies how the content delivery path is constructed during the resolution phase. Finally, *security* and *scalability* issues on this approach are discussed

In Section 5, server-awareness techniques in the COMET system are presented. In particular, in this project we assume that, more often than not, multiple copies of a content are available throughout the Internet. This may happen due to several reasons, the most common of which is replication of content in CDN surrogate servers. Therefore, during the resolution and before the actual delivery of content, the most appropriate server has to be chosen. The discussion on this issue as well as our first thoughts has been initiated in Deliverable D3.1. In this document, we have elaborated further the specific techniques that have to be integrated into our architecture, as well as on the specific messages that need to be exchanged between the related entities on run-time. That is, for instance, server-load (as well as network load) changes dynamically in the network and therefore, a server that was marked as “suitable” due to “low” load may suddenly become loaded and not fit for purpose anymore. In that case, the corresponding “UPDATE” messages have to be exchanged between the involved entities in order to alter the target source of the content. The specifics of these operations are discussed in detail in Section 5 of the present document.

Server-awareness has to be coupled with knowledge on network conditions. That is, even if we find the most suitable server in terms of server-load, the conditions of the path leading to this server have to be investigated too. If the path experiences high load, or congestion at some point, then maybe the tuple *server-path* will not be able to guarantee the Quality of Experience we want for the end-user. Having said this, in Section 6, we present the decision strategies for CAN; these strategies are based on objective functions to decide which of the available *server-path* combination is the most suitable for the content delivery.

## 2 Introduction

The transition from a host-centric Internet to a content- or information-centric one requires major changes to several vital components and operations of the Internet. In a content/information-centric Internet, contents are resolved based on their name instead of the name/address of the machine that hosts them. Therefore, a content's name plays a vital role in the dissemination and consumption process.

In this document, we provide the specifications for the two resolution and delivery approaches of the COMET system, namely the Decoupled and the Coupled approaches. In the Decoupled approach, content resolution is a different process to content delivery. This approach is based on the evolution of the current DNS system. In contrast, the Coupled approach comprises a more revolutionary approach, where pointers to the actual content can be found in specific individual domain. Furthermore, in the Coupled approach, preparation for the content delivery process commences during the content request and resolution process. Therefore, the resolution process is based on hop-by-hop discovery of the requested content, during which, the return path for the content delivery is setup to speedup content delivery. Clearly, this approach is closer to the *content-centricity* notion of the future Internet.

Obviously, both approaches have implications to the content publication and content request procedures. For example, in the Decoupled approach, a central entity needs to keep records of the published contents, while in the Coupled approach the content search spreads through the network gradually, i.e., if one domain does not have the required info to reach the content, then the request is forwarded upwards to the higher-tier domain. This prerequisites that the publication process has already happened in a hierarchical manner, where each domain-hierarchy is responsible/aware of published contents.

We discuss the final specification of these processes for the Decoupled and the Coupled approach in Sections 3 and 4 of the present document, respectively.

However, for the system to provide increased quality of experience to the end-users, the content discovery procedure has to be complemented with efficient content delivery mechanisms. That is, given the evolution of CDNs, in COMET we assume that more often than not, multiple copies of a content will be available. This inherently implies that multiple return paths (i.e., from the server to the user) can be used to fetch the content back to the requesting user. The selection procedure has to take into account two main factors: i) the server load, mainly in terms of numbers of concurrent requests, and ii) the path conditions, in terms of load, delay and congestion. That said, in COMET, we have designed “Server-awareness” algorithms, to prevent requesting the content from heavily-loaded hosts as well as “Path Selection Strategies”.

Our “server-awareness” designs are discussed in Section 5, where we detail the related mechanisms both for the Decoupled approach, where server monitoring and selection is independent of the content resolution process, as well as for the Coupled approach, where, by definition, server-selection is made during the resolution process. The combination of server *and* path selection, in order to choose the most efficient tuple is discussed in Section 6. To assess the credibility of the best server to download the content from, taking also into account the paths that lead to that server, we use Objective functions. The parameters here are combinations of: i) load of the available servers, ii) path length (shortest or not), and iii) best path (in terms of delay and congestion).

A content- or information-centric Internet will inherently support caching content *in the network*. Hence, one may argue that server-awareness is going to become less of an issue, once full deployment of in-network caching is in place. However, given the tremendous increase in user-generated content the last few years, as well as the expectations for future growth of Internet traffic, the sustainability of in-network storage will have to be questioned. That is, although memory is becoming much cheaper than bandwidth, or deployment of server farms, it is highly improbable that all Internet content will be stored/cached inside the network. Therefore, although popular content will be accessible from in-network caches, the notion of surrogate servers or CDNs



will still exist. Having said this, our designs for location independent content resolution, routing and server and path selection will still be valid in a future content-centric Internet, where in-network caching will be part of the core Internet infrastructure. In this context, we are also investigating efficient mechanisms for in-network caching, which will enhance network-wide performance. Our solutions and findings on these aspects are reported in D4.2 [1].

## 3 Decoupled Approach Specifications

### 3.1 Overview

This section contains the specifications for the CMP in the Decoupled Approach, which advances the current Internet system, inheriting the required functionalities along with content-based enhancements. It is developed with less disruption to deployment in view. It follows a more natural evolution path of the current Internet system towards a content-centric one. The look-up based paradigm employed with its foundation built upon the current DNS system would allow gradual deployment with minimal disruption.

### 3.2 Entities and Interactions

According to the architecture defined in D2.2 [2], the decoupled approach consists of the following entities

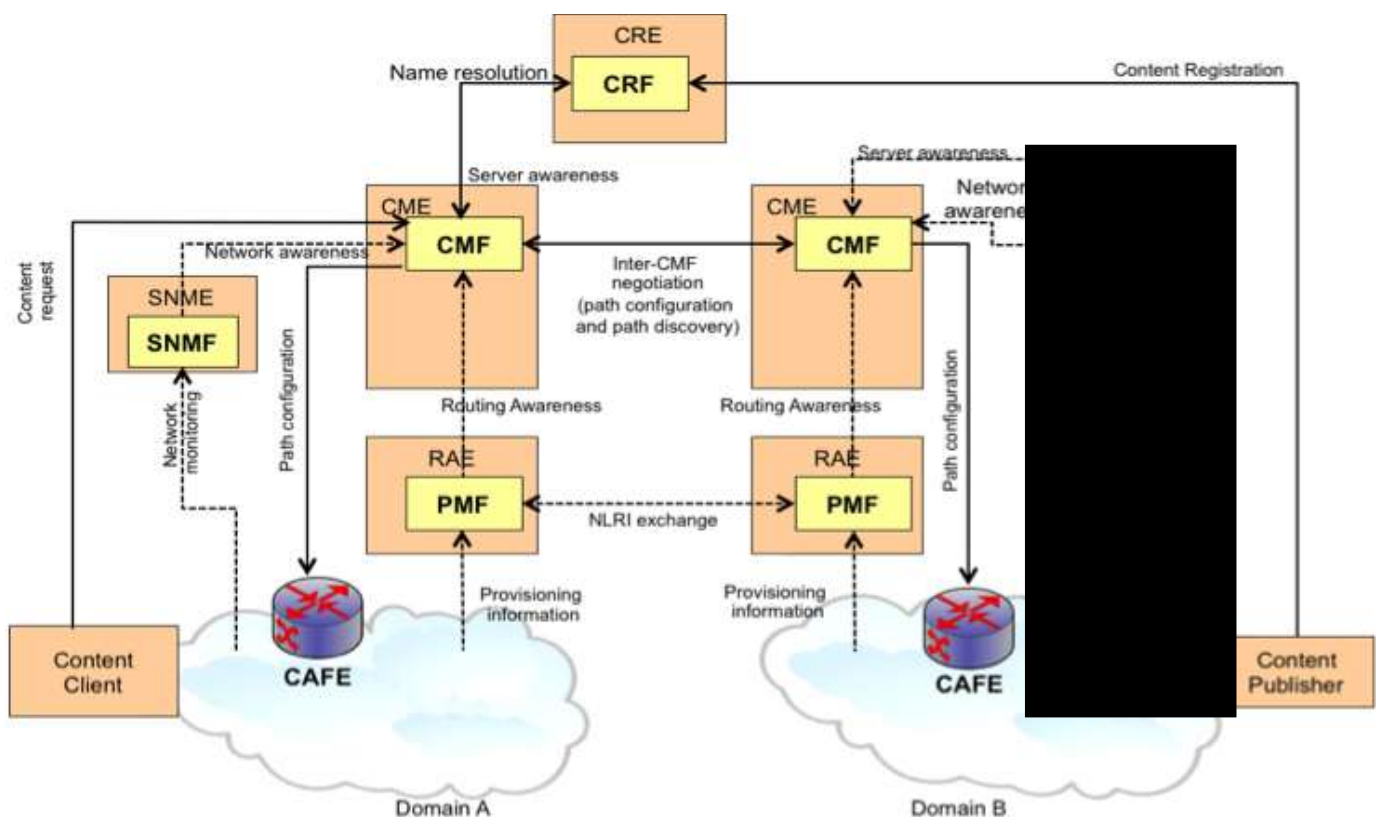
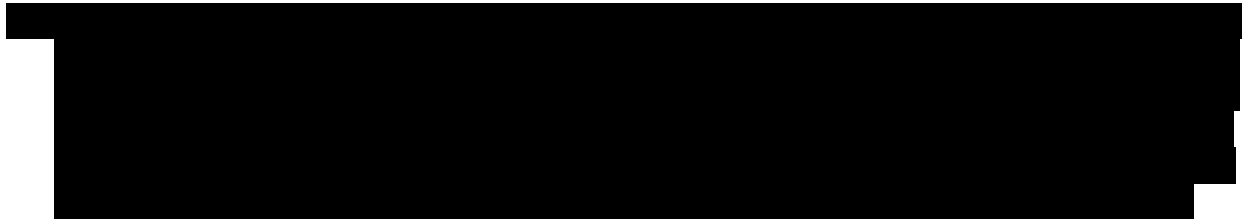


Figure 3-1: Diagram of Entities

- Content Client (CC), enables the user to request the retrieval of a specific content identified in Comet by its Content Name or Content Identifier and then launches the suitable application for playing the content, when the characteristics of that Content are received back.
- Content Publisher (CP), enables a Content Owner to publish information about a specific content to be distributed by Comet and stores the relevant pieces information in the format of Content Record.
- Content Resolution Entity (CRE) keeps track of Content Records and resolves Content Names to Content Record(s).



- Route Awareness Entity (RAE) provides network reachability info to the CME. Its functionality, together with its interface to CME will be explained in deliverable D4.2 [1] because of its close relation with the Forwarding Plane.
- Content Mediation Entity (CME) gathers information from RAE, SNME and CRE, makes decisions on the most suitable path for reaching a specific content and configures the delivery path for retrieving this content. Besides exposing an interface to every element enumerated above, an inter-CME one is required to obtain information about the servers from the client's CME and ensure the set-up of a tunnel between the CC and the Content Server (CS).
- CAFE delivers the content through the paths set up by the CME, its specifications are explained in deliverable D4.2 [1] since this entity is located in the Forwarding Plane.

It is important to remark that while CME, SNME, RAE and CAFEs are located inside each ISP/domain, CREs are “outside” the ISPs, hence the decoupled name. Therefore, a single CRE can manage several ISPs and there exists a hierarchy of CREs (DNS-like), so that a root CRE can be queried in order to find out which CRE stores the Content Records for a Content Name. Besides, there will be always more than one CAFE per ISP, typically located at the links connecting the ISP with other ISPs and at the connection points of CSs and CCs.

Each Entity can be divided into Functional Blocks as depicted in the following figure, together with the interfaces between them (in this section, we describe in detail the blocks included in CC, CME and CRE and provide a brief overview of the SNME). Since a Client's ISP has to retrieve information from the Server's ISP, the interfaces joining the respective CMEs and those elements triggered by those inter-domain operations have been also depicted, although obviously both CMEs implement the entire set.

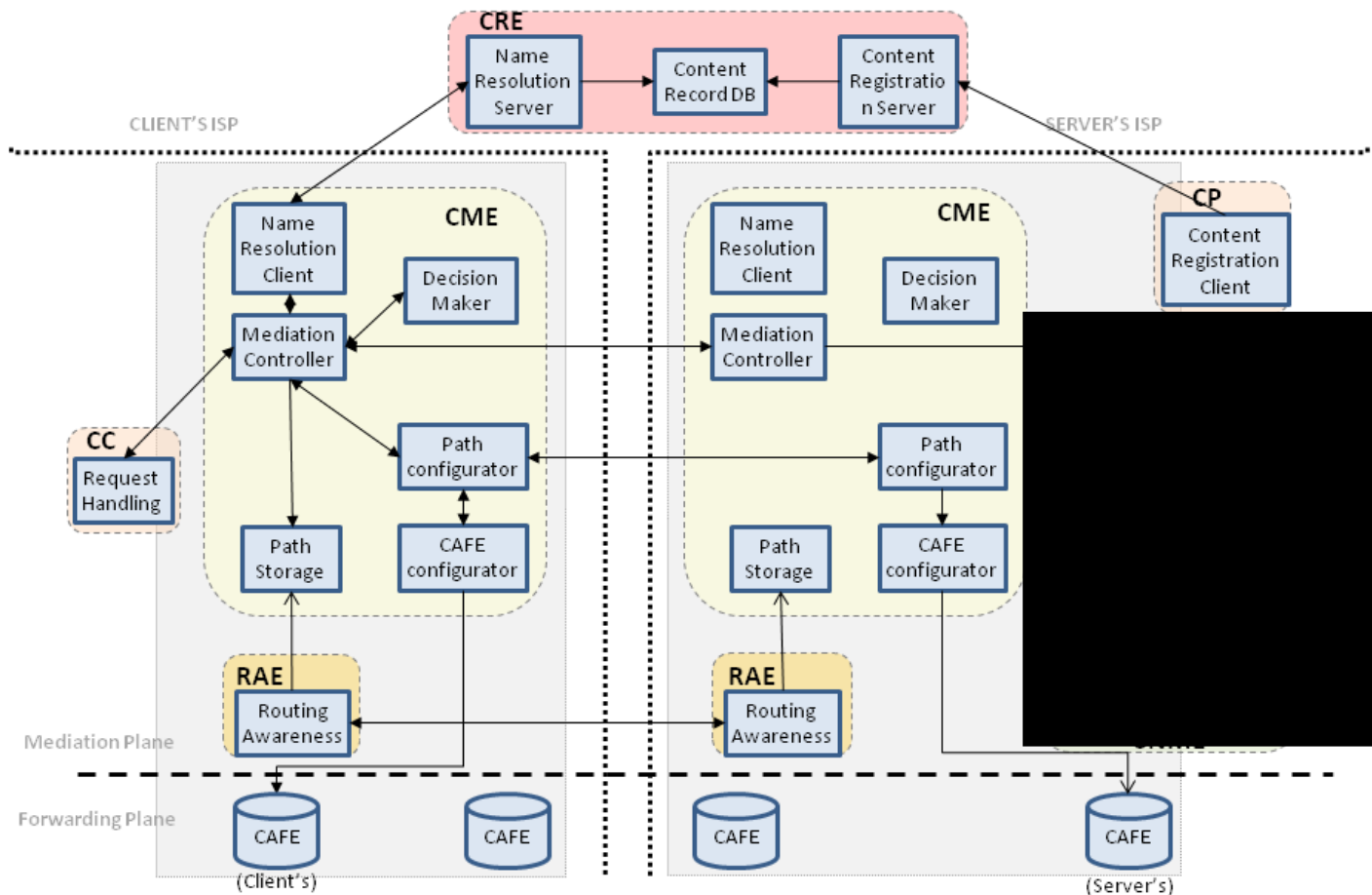


Figure 3-2: Functional Blocks

- For the CC, there is only a functional block, the Request Handling (RH) that sends the Content Name/Content ID to the Mediation Controller (MC) at the CME, and processes the information send back by the MC, once a path to a destination server has been found
- For the CME, the following functional blocks are involved
  - The Mediation Controller (MC) acts as a front-end to the CC and organises the flow of information inside the CME, exposing internal interfaces to most of CME's internal blocks, besides communicating with other MC at remote CMEs in order to retrieve information about CS's load. A third interface enables a remote MC to query its domain's SNME.
  - The Name Resolution Client (NRC) queries the Name Resolution Server (NRS) at the CRE in order to resolve Content Names and Ids into Content Record. An internal interface links the NRC with the MC.
  - The Decision Maker (DM) decides on the most suitable path to retrieve a specific content. An internal interface links this functional block with the MC
  - The Path Storage (PS) stores the routes that lead from one ISP origin to an ISP destination, expressed in AS's list format. These routes are fed by the RAE, whose interface with PS is described in D4.2 [1]. An internal interface links the PS to the MC.
  - The Path Configurator (PC) translates the Path defined as lists of AS into a path defined as a list of network elements (CAFEs). To carry out this translation, it queries the PC in the server's CME for information about the network elements involved in the path. Apart from the interface linking the PC with the MC, a third interface connects the PC with the CAFE Configurator (CaCo) to allow the configuration of CAFEs with the translated list of network elements.

- The CAFE Configurator (CaCo) instructs the Client's attached CAFE and the Server's attached CAFEs with list of network elements that have to be crossed in order to travel from the CC to the CS and backwards. The interface connecting the CaCo with the CAFEs is described in D4.2 [1].
- The CRE can be split up into
  - Name Resolution Server (NRS) that manages the queries from the Name Resolution Client in the CME and returns back the Content Record(s) stored in the Content Record DB
  - Content Record DB (CRDB) that stores the Content Records describing the contents.
  - Content Registration Server (CRS) that stores in the CRDB, the Content Records sent by the Content Registration Client (CRC) at the Content Publisher (CP).

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

- Finally, the Content Publisher (CP) contains a Content Registration Client (CRC) that sends to the Content Resolution Server (CRS) at the closest CRE the information concerning a specific content and the Content Servers distributing them in order to store them in the CRDB as a Content Record

The procedures performed within the Content Mediation Plane basically are:

- Content Publication Process, where a Content Owner publishes in the CRE information about a specific content and the Content Servers distributing it. This procedure is described in section 3.3.
- Content Resolution Process, where the CC receives information from the CM at CME about the Content Server distributing a specific content and the protocols for retrieving it, after having requested a Content Name or Content Id. This procedure is described in section 3.4.
- Server Awareness, where the SNME gathers the status of the different Content Servers (CS) deployed in its ISP. This is described in section 5.
- Routing Awareness, or how an ISP learns the path to other ISPs, is explained in deliverable D4.2 [1].

Nonetheless, a brief description of *Content Publication* and *Content Resolution* is provided here as a general introduction.

The *Content Publication* procedure stands for the operations that allow a Content Owner to make a specific content available throughout Comet by means of publishing Content Records to CRE. A Content Record typically stores the following pieces of information, as illustrated in the following figure.

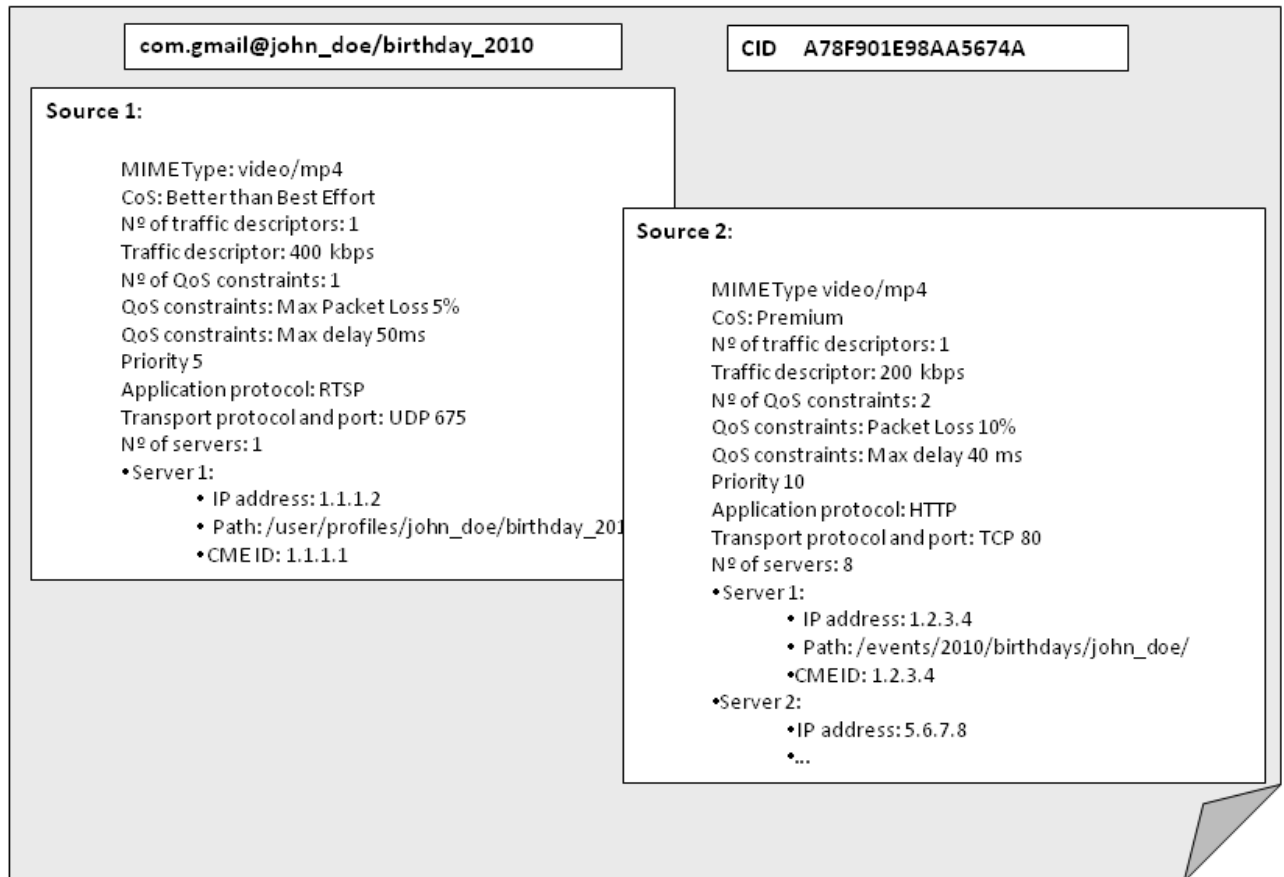


Figure 3-3: Content Record

- The Name of the Content which appears in two flavours:
  - One Human oriented, the Content Name, that has the structure domain@owner/path like (com.gmail@John/mySong), and is specified by the Content Owner.
  - One Machine Oriented, the Content Id, created by the CRE and used internally across the CMP.
- A list of Content Sources serving the content, which differ in
  - Mime Type identifying the type of content
  - Comet Class of Service (CoS) that can take the values: Premium, Better than Best Effort, Best Effort.
  - Traffic Description (Bandwidth)
  - QoS Constraints (Packet Delay, Packet Loss)
  - Application and Transport Protocol
  - Priority of the Source
  - List of actual Content Servers distributing the content, identified by IP, Path and CME ID where they are located

The Following figure depicts the entities and functional blocks involved in *Content Publication*, as well as the data flows among them.

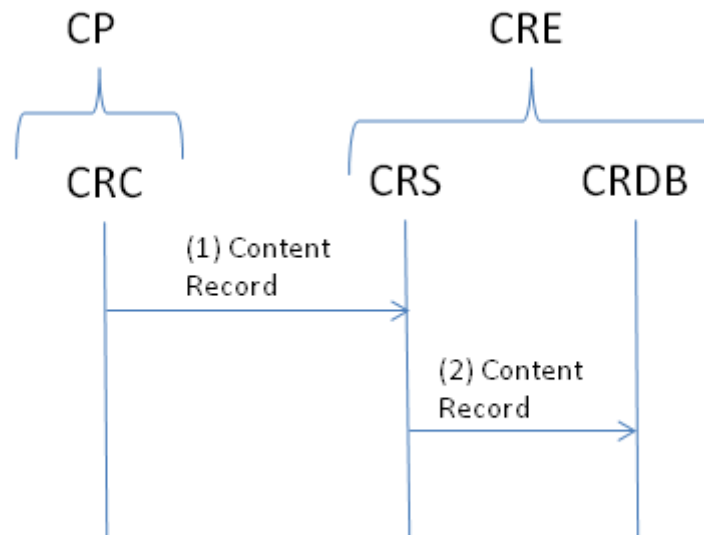


Figure 3-4: Content Publication

**The sequence of operations is as follows**

1. CRC at CP sends (1) a registration query to CRS at CRE. The chosen CRE is the one in charge of managing the domain where the content will be registered to. The result of this operation is a Content Record.
2. CRS stores (2) the Content Record in CRDB.

The *Content Resolution* Procedure consists of all the operations performed by the CMP to resolve a Content Name and instruct the CC with the means to reach a Content Server distributing it. They are briefly described in the following figures and explanation:

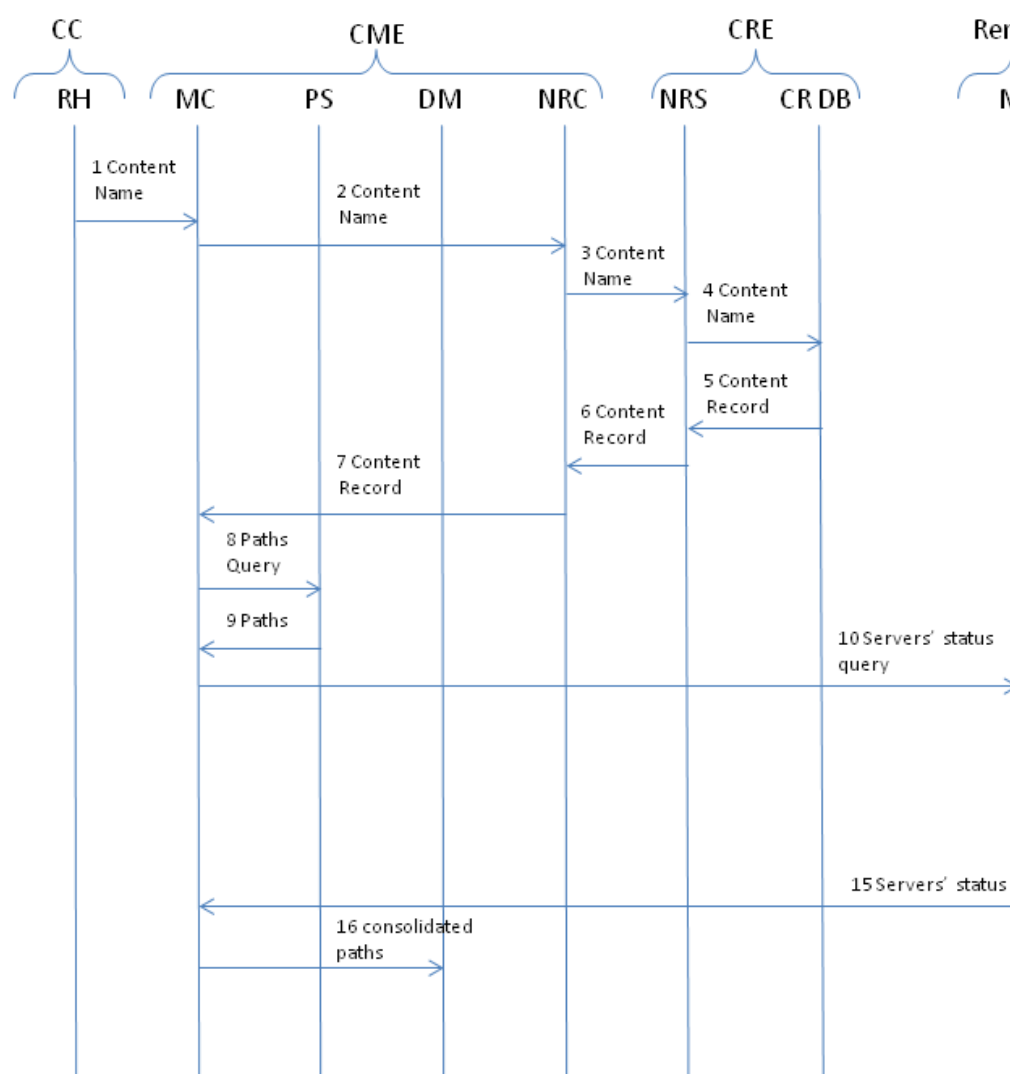


Figure 3-5: Content Resolution Process, first stage

The first operation is called *Content Request*, where a CC requests the retrieval of a specific content.

1. RH at CC sends a query (1) MC to its local CME, carrying either the Content Name (human readable) or the Content ID (machine) of the requested content.

The second operation is called *Name Resolution*, where the Content Record for the requested content is retrieved from the CRE.

- MC at CME sends a query (2) to NRC at CME to resolve the Content Name/CID
- NRC at CME forwards the query (3) to NRS at CRE. Since CREs are “outside” the ISP, NRC first has to find out which CRE can serve its query
  - Either the Content Domain CRE, if this is already known by NRC.
  - Or Root CRE if NRS is unaware of the CRE managing the queried domain. Root CRE then redirects NRC to the right CRE.
- NRS looks up (4) the queried Content Name in CRDB
- The Content Record is retrieved (5) from CRDB.
- NRS at CRE forwards (6) the Content Record to NRC at CME.



- NRC at CME forwards (7) the Content Record to MC.

At this moment, MC has at its disposal a Content Record containing a list of Content Sources grouped by type of Content, Priority, Transport and Application protocols, Traffic and QoS constraints and actual Content Servers providing the content.

These sources are ordered according to CoS and Priority, first those sources matching the client's CoS and then those with lower ones (Premium > Better than Best Effort > Best Effort) discarding those sources with higher CoS than client's. For sources with the same CoS, the ones with higher priority are placed first. Therefore, the following operations (*Path Discovery*, *Server Awareness* and *Decision Process*) are first applied to the source matching Client's CoS and with higher priority. If a valid solution for path/server is not found, the next source in the ordered list is processed until a valid solution is found or the list is emptied.

The Next operation is *Path Discovery*, where the paths to each Content Server specified in the Content Sources are found out.

- MC sends a query (8) to PS to find out the known paths to the ISPs of each server included in Content Record
  - For paths not stored in PS yet, MC at client's CME sends a query to MC at server's CME to retrieve this information, according to the *reverse approach* (not depicted in the figure). In this approach, the remote CME will obtain the path from its PC and send it back to the client's CME.
- The paths are retrieved (9) from PS

Now, MC has added to the previous information a list of paths to each Content Server specified in the Content Source, qualified with traffic/QoS parameters. Each path consists of the sequence of AS IDs that have to be crossed to reach the target ISP and the Traffic/QoS parameters used for qualifying it. Those parameters are the provisioned bandwidth of the path, its packet loss and its packet delay.



At this point in time, MC at the client's CME has gathered information about Content Sources, Qualified Paths to them and Servers' Status.

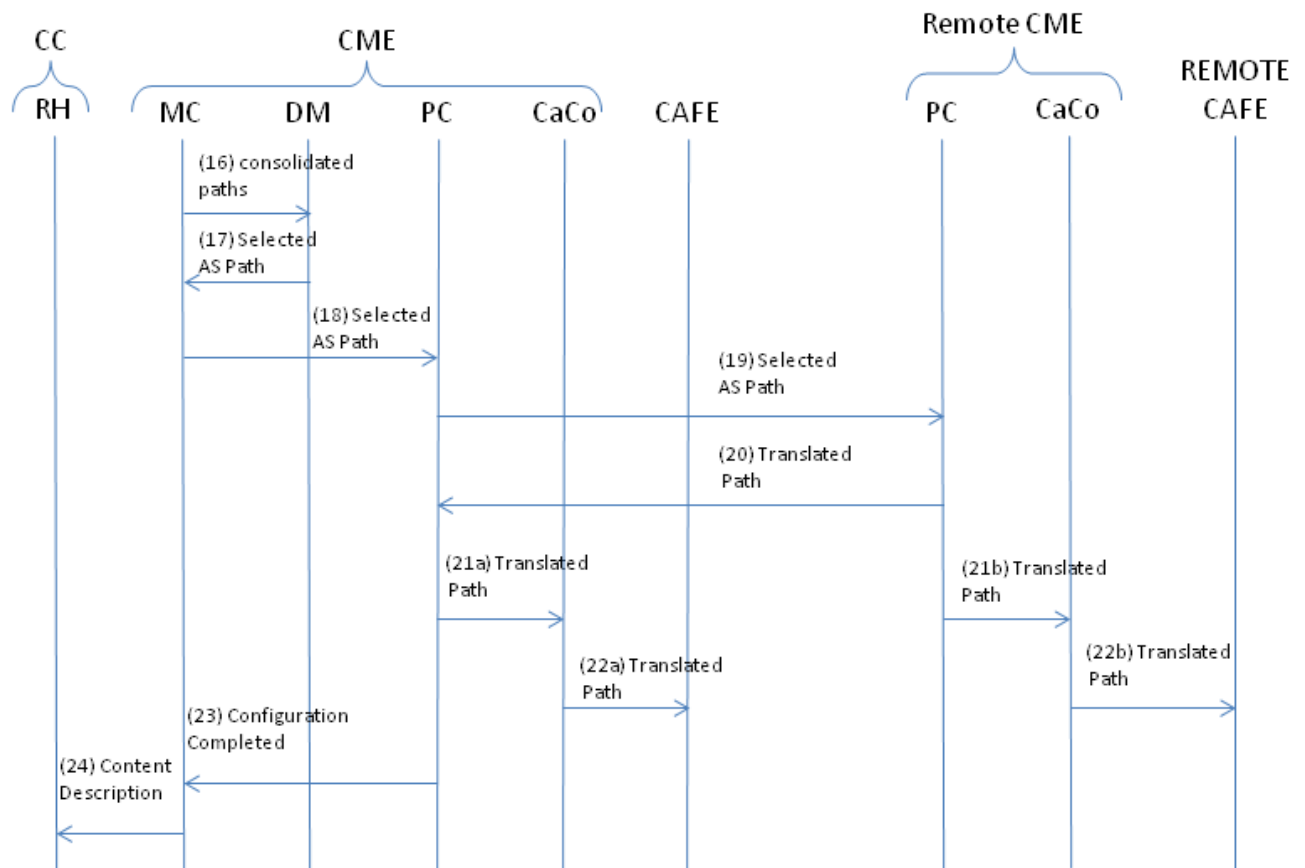


Figure 3-6: Content Resolution Process, second stage

Now it is time to launch the *Decision Process*, where an optimal path for Content Retrieval will be selected by using the information gathered up to this moment

- MC sends (16) the consolidated information to DM
- DM which weighs up the possible paths, and returns back (17) an optimal solution.
- MC sends (18) the information of the selected path to PC. At this moment, the path is still described as a list of AS Ids, that has to be translated into a list of network elements, the CAFEs in each ISP configured to forward packets from that ISP to the next one in the list plus the content server's attached CAFE and the client's attached one.

If a valid solution has been found, the next Operation is *Path Configuration*, where both client's and server's CME instruct the CAFEs attached to the Content Server and the Content Client with the list of network element to be traversed forward and backwards

- Client's PC sends (19) the Server's PC, the path in AS ids format together with info about the Client's attached CAFE and the local CAFE connecting to the next hope in the AS list.
- Server's PC answers (20) with the translation of the AS list plus the server's attached CAFE.
- Both Client's and Server's PC inform (21a) (21b) their CaCos of the translated list of network elements.
- The local and remote CaCos respectively configure (22a) (22b) the Client's attached CAFE and Server's attached CAFE with the list of network elements to be traversed.
- Local PC reports (23) the MC that the configuration has been accomplished.

- MC sends (24) to the RH at CC the features of the server (IP address/port, transport and application protocol, Mime Type of the content) the client has to connect to retrieve the content.

Next sections explain in fuller detail each of these procedures and functions.

### **3.3 Content Publication Process**

Content Publication is the process of making content available to Content Clients and includes two sub-operations, content storage and content registration.

1. Content Storage is the sub-operation of storing content to the content servers. It is performed either from the content owners or providers (large organizations or end-users) storing their content in their own premises (content servers or hard disk) or from large content providers (e.g. Youtube, Facebook, etc.) with which small content owners have made an agreement to store their content in content providers' content servers. In any case, the content owners or providers have to collect all required information and metadata to be included later in the content record. However, this sub-operation is out of scope for the COMET project.
2. Content Registration is the sub-operation of creating and storing the content record for the associated stored content into the CRE, including content's required information and metadata.

It is assumed that the content owner has acquired part of the global COMET namespace (naming authority) from a registrar. This is necessary only once and is similar to the acquisition of a domain name in DNS. The registrar will typically provide the Content Owner with an account on a local CRE to create and administrate content records associated with the allocated namespace, while information about the naming authority and its serving local CRE are stored in the root CRE.

Every content with a given content name is associated with a content record. An example of a Content Record is shown in the next figure.

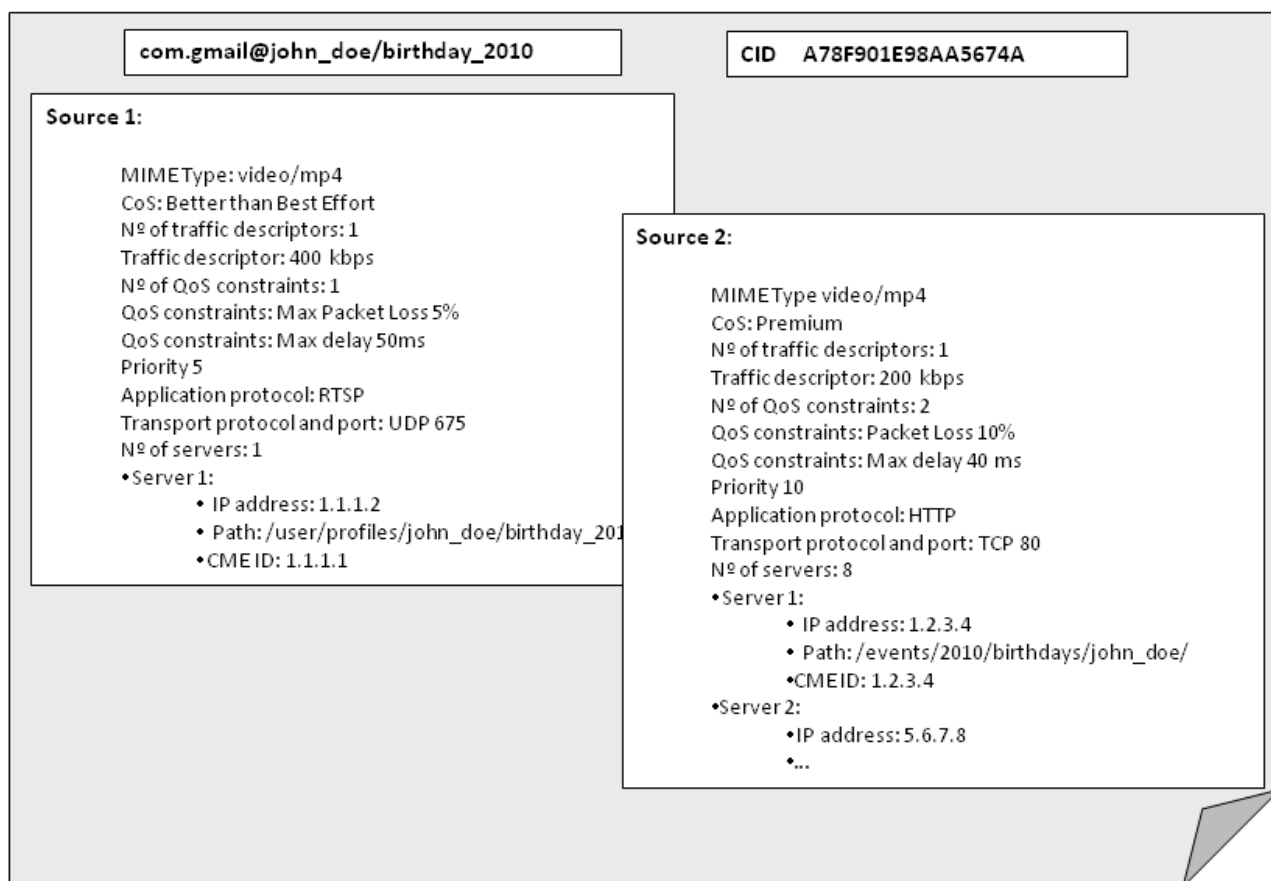


Figure 3-7: Example of a Content Record

In general, a Content Record consists of the following information:

- Content Name (unique, human-readable, alphanumeric string, main identifier of the content) that will be typically specified in the format `domain@owner/path`.
- Content ID (unique, human-unreadable, alphanumeric string, one-to-one mapped with the content name, which is used as an alternative identifier of the content)
- List of Content Sources, consisting of:

- MIME type (e.g. audio/mpeg, video/mp4, etc.)
- CoS (BE or BTBE or PR):

The concept of CoS is related to the Service Level subscribed by the user. A content owner, when defining sources for its contents, can specify which sources should be reserved for clients with a precise CoS. The actual requirements of the source, in terms of QoS, are defined by the two next parameters, decoupling User SLA from the content characteristics specified by the content owner. In other words, each Content Owner can define its own QoS constraints per CoS for the contents it distributes.

- QoS constraints (IP Packet Transfer Delay [msec] and IP Packet Loss Ratio)
- Traffic Descriptor (Peak Bit Rate [kbps])
- Priority (1-10)

This parameter is used for creating a hierarchy among sources. For instance, a piece of content can be distributed by means of Streaming/P2P/Direct

Download sources. It seems sensible that the Content Owner prefers that at first users are directed to the Streaming Server, and then when the Streaming Servers are at full capacity to the P2P, and finally to the Direct Download Ones. The only mean to implement this functionality is the use of priorities to decide which sources should be used first. Other example would be the case of a Content Owner that wants to distribute its contents first by streaming and then by VoD, what could be carried out easily by changing the priorities of the sources on the fly.

- Application/Session protocol (e.g. HTTP, etc.)
- Transport protocol (e.g. TCP, etc.)
- Transport port
- List of Content Servers:
  - IP address
  - Server path (the path where the content is located inside that server, e.g. /content/videos/user/myvideo.mp4)
  - CME IP address (the IP address of the CME adjacent to the content server)

Content owner creates and administrates content records through the Content Publisher. The Content Publisher exposes both a web-based user interface and an API for this purpose. This means that any web browser can be used to publish content in COMET. The API permits integration with existing content publication platforms, such as YouTube and Facebook, for instant and automatic publication to COMET.

Content registration sub-operation follows the Handle System protocol and processes [4] [5] and is presented in the following sequence diagram (Figure 3-8) and the respective steps below.

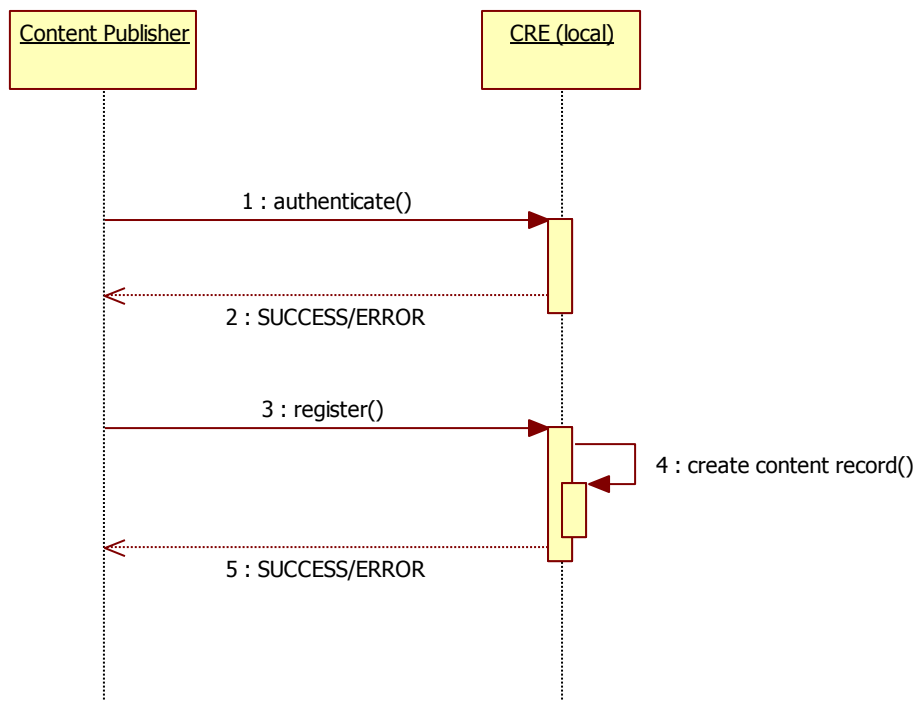


Figure 3-8: Content registration sequence diagram

1. Content owner sends an authentication query to the CRE through the Content Publisher, including <Naming Authority, Password>

2. Content publisher receives a SUCCESS or ERROR message from the CRE, confirming/declining content owner's access to the content record management process.
3. In case the authentication was successful, a "register" message is sent to the CRE, consisting of <CREATE, Content Record>.
4. CRE receives content record parameters, creates and stores the content record in its local database.
5. Finally, content publisher receives a SUCCESS or ERROR message confirming the result of the content registration. In addition, content owner is also provided with the generated Content ID.

Content owner is also given the ability to administrate (update or delete) its content records when it is required, through the same interface (Content Publisher – CRE). The content record administration process is similar to the one described above and is presented in Figure 3-9:

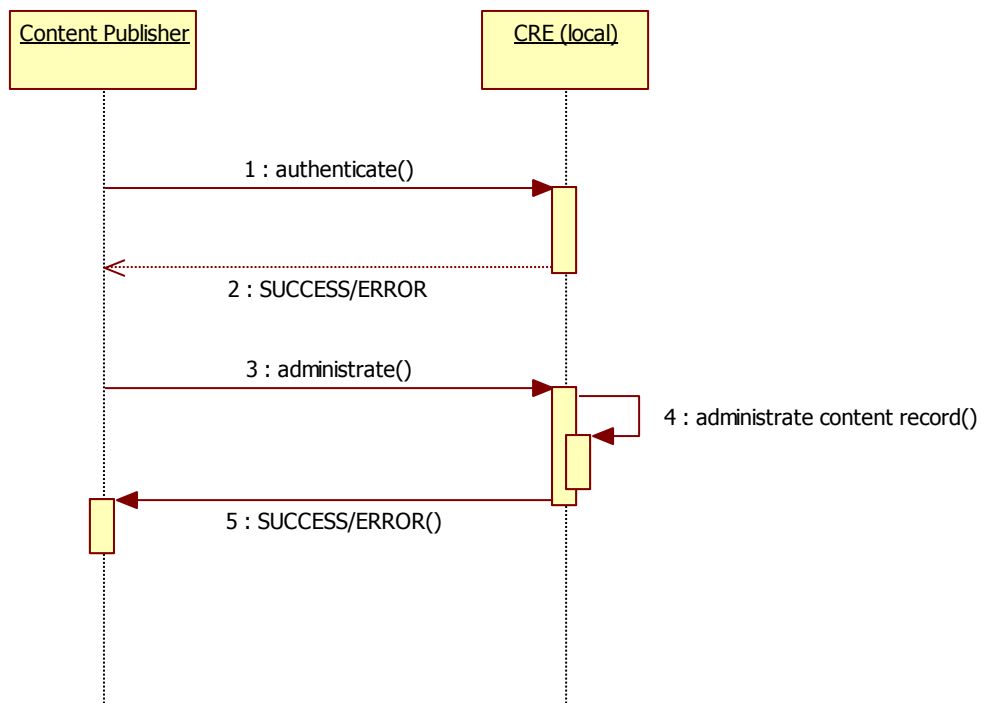


Figure 3-9: Content record administration sequence diagram

1. Messages 1, 2 and 5 have the same structure and information as the ones described in content registration. The main difference is that the "administrate" message has the following structure: <ACTION, Content Record>, where ACTION could be either UPDATE, in case of content record parameters update or DELETE, in case of permanent content record deletion.
2. The content registration and administration processes can also be performed in batches, meaning that the content owner will be able to prepare a file with all desired operations (multiple content records creation, update and deletion), which could be uploaded and processed in the Content Publisher.

## 3.4 Content Resolution Process

### 3.4.1 Content Request

This process is done when a Content Client requests a particular content to the CME. In this step, the Content Client sends a Content Name or a Content-ID in order to express his purpose of retrieving a particular content. At this point, we assume that an end-user has found the CN or the CID of the content he wants to consume, either by searching it through keywords in search engines or by obtaining in alternative ways (e.g. via email).

After this message, the CME will start the Content Resolution process, which is explained in the next section.

The process of the Content Request is depicted in the following figure:

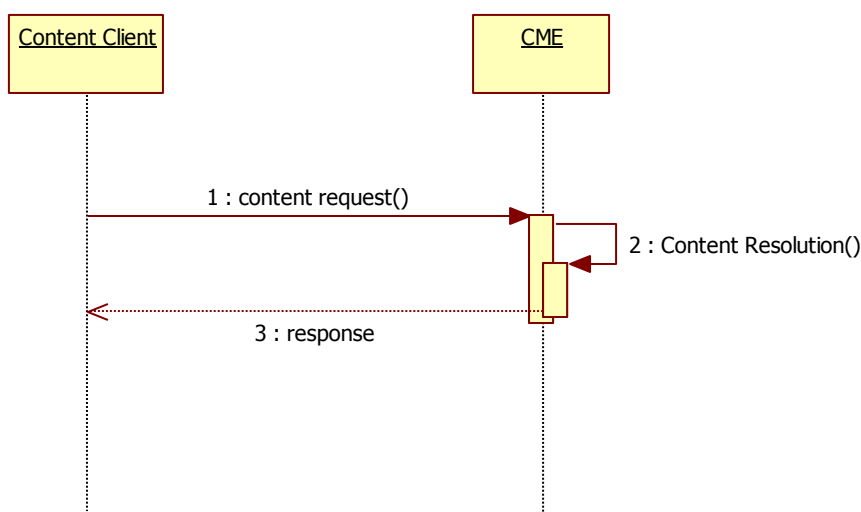


Figure 3-10: Protocol between the Content Client and the CME

The lists of parameters that are exchanged through the CC-CME interface are the following and are more detailed below:

Content request (query)	Response (answer)
<b>QNAME (CNAME or CID)</b>	<ul style="list-style-type: none"> <li>• <b>CID</b></li> <li>• <b>MIME type</b></li> <li>• <b>Application/Session protocol</b></li> <li>• <b>Transport protocol</b></li> <li>• <b>Server Port</b></li> <li>• <b>Server IP length</b></li> <li>• <b>Server IP address</b></li> <li>• <b>Path</b></li> </ul>

Table 1: Parameters in a Content Request

#### 3.4.1.1 Basic Procedures

For the execution of the Content Request, the Content Client needs the following configuration parameters:

- IP and port of the responsible CME.

- Content Name of the particular content to be consumed. (Supposed to be known by the Content Client).

When the CC sends a Content Request carrying a QNAME and in order to avoid waiting indefinitely for the CME response, a timeout is defined for the client. When this period is elapsed, the client should send the Content Request again. This timeout should be hard-coded for the COMET Content Clients. After two retransmissions, the Content Client should give a no response error message to the application. If CME was able to resolve the content Name and locate a server that could provide the content to the client, a response will be received with the information enumerated in Table 1.

The general structure of the query message can be found in annex B.1.1, while the structure of the response message is specified in annex B.1.2.

The COMET Content Client must be able to access contents using the most common Application Protocols (http, ftp, rtsp, and so on, as described in Table 13 in annex B.1.2), and choose the system application which will open the content.

The process to assign an application protocol with the correct application will require setup actions from the user during installation or execution of the COMET Client, which will be different depending on user's OS. The rationale behind this decision is that COMET does not support or impose the use of any kind of media player or similar software. The user is free to choose their usual programs (VLC, MPC, Winamp, WMP, etc.), thus requiring specific configuration steps from the user.

E.g. in Windows during the installation, protocols are added at Windows registry. When a user accesses the content, he/she will be asked for the application to open it. For Linux/Unix OS it has been considered to use configuration files.

About MIME Type is an interesting indicator when the content is stored at the hard drive but the priority must be for the Application Protocol. Content with the same MIME type can be required with many different protocols (mainly http, ftp or mail) and when the user received the content the related program is launched. But as the same way that when an HTTP request is launched and the system open the navigator (independent of the MIME type of the content required), when a RTSP request is launched the system must start the application which 'understand' the RTSP protocol (again independent of the MIME Type that the content have).

### 3.4.2 Content Resolution

Content Resolution is the process of resolving the received content name to the associated content record stored in the CRE. The operation is performed in the CME, upon reception of a content request from the Content Client and is handled by the Resolver component within the CME, which interacts with the CREs. Content Resolution follows the Handle System resolution operation in a similar manner. More specifically, the process is presented in the sequence diagram of Figure 3-11 and also described in the respective steps below.



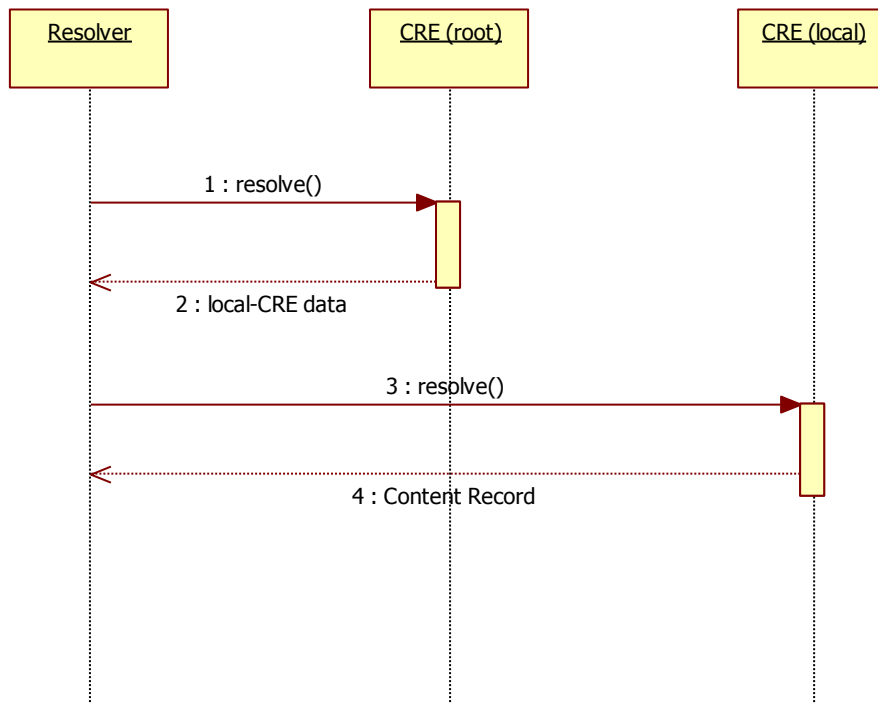


Figure 3-11: Content Name resolution sequence diagram

1. Resolver extracts the naming authority part of the received content name and checks its local cache if information about the local CRE responsible for the requested naming authority already exists. If not, then sends a `resolve(NA)` message to the root CRE. Root CRE information is configured in CME's database, through the CME administration interface. Otherwise, resolver skips this step and goes to step 3.
2. Root CRE returns available information for the local CRE serving all requests for the specified naming authority (if it exists to root CRE database), and resolver saves the received parameters to its local cache for a specific period of time. In case that naming authority does not exist, an ERROR message is delivered and content resolution is terminated.
3. Resolver sends a `resolve(CN)` message to the discovered local CRE to receive the respective content record
4. Finally, the content record is returned to Resolver (if content name exists to CRE database), otherwise an ERROR message is delivered.

### 3.4.3 Path Discovery

The objective of Path Discovery is to retrieve paths going from given server towards the client. The information about paths is stored in the Path Storage component of CME. It maintains following types of data:

- Type 1: Information about local paths and their parameters , i.e., COMET CoSs, values of path parameters, i.e. IPTD, IPLR, max BW, from one edge of the domain to another edge of the domain (edge can be defined by an AS number or a prefix). In case of access domains (domains with clients or with servers), this covers: 1) complete paths inside the domain (server and client in one domain), 2) fragments of paths from edge to the client, and 3) fragments of paths from server to the edge. This information is provided by the RAE.
- Type 2: Information about inter-domain paths originating from the edge of local domain and ending in given prefix (for given COMET Class of Service). These paths are provided by the RAE.
- Type 3: Cached paths originating from given prefix and ending in another prefix (for each COMET Class of Service). These paths can be provided by: 1) Mediation Controller (reverse

path retrieval), 2) Path Storage using assumption of path symmetry, or 3) static configuration. Those paths may have a limited life time if they are not static.

Path Storage provides following interfaces:

1. `retrievePaths(clientIp, serverIp, classOfService)`. This interface is used by Mediation Controller to get a list of paths regardless of their Type.
2. `updatePaths(classOfService, paths)`. This interface allows Mediation Controller to update particular path with dynamic Type 3 information.
3. `updatePathsStatic(classOfService, paths)` This interface allows Mediation Controller to update particular path with static Type 3 information. This request may arrive from CME administration interface.
4. `updateProvisioningRAE(.)`. This interface is used to update the Type 1 information, which is provided by RAE.
5. `updatePathsRAE(.)` This interface is used to updates the Type 2 information, which is provided by RAE. See RAE-CME interface specification

### 3.4.3.1 Message sequence diagrams

Figure 3-12 presents the message sequence diagram for the most frequent behavior during Path Discovery when path is cached in the local Path Storage.

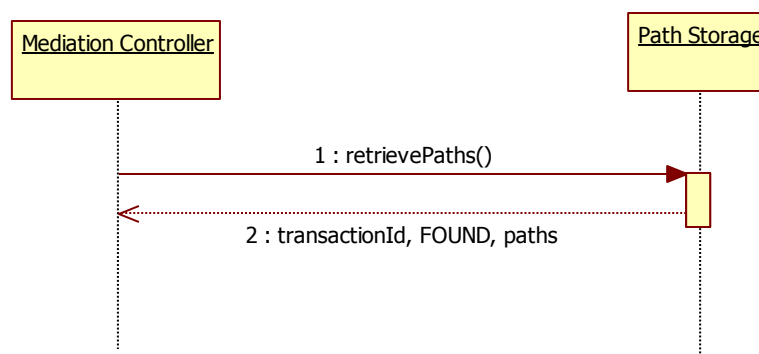


Figure 3-12: Path Discovery sequence diagram for cached paths.

Description of the messages:

1. `retrievePaths(transactionId, clientIp, serverIp, classOfService)`

If the `clientIp` and `serverIP` addresses belong to local domain, the Path Storage performs longest prefix matching using Type 1 data for given *classOfService*. If there is no path between them, the result NOTFOUND should be returned. On the other hand, if `serverIP` addresses does not belong to local domain, Path Storage performs longest prefix matching against the Type 3 data for given *classOfService*. If there are no paths between them, the result UNKNOWN should be returned.

2. `transactionId, FOUND, paths`

In this diagram we assume that matching is successful (status=FOUND) and paths are returned:

- a list of path information structures, where each structure has:
  - AS path as a sequence of AS numbers,
  - Source prefix is related to `serverIP` address
  - Destination prefix is related to `clientIP` address
  - QoS parameters: IPLR, IPTD, maximum BW for a single delivery,

Figure 3-13 presents the sequence diagram when path is not found in the local Path Storage.

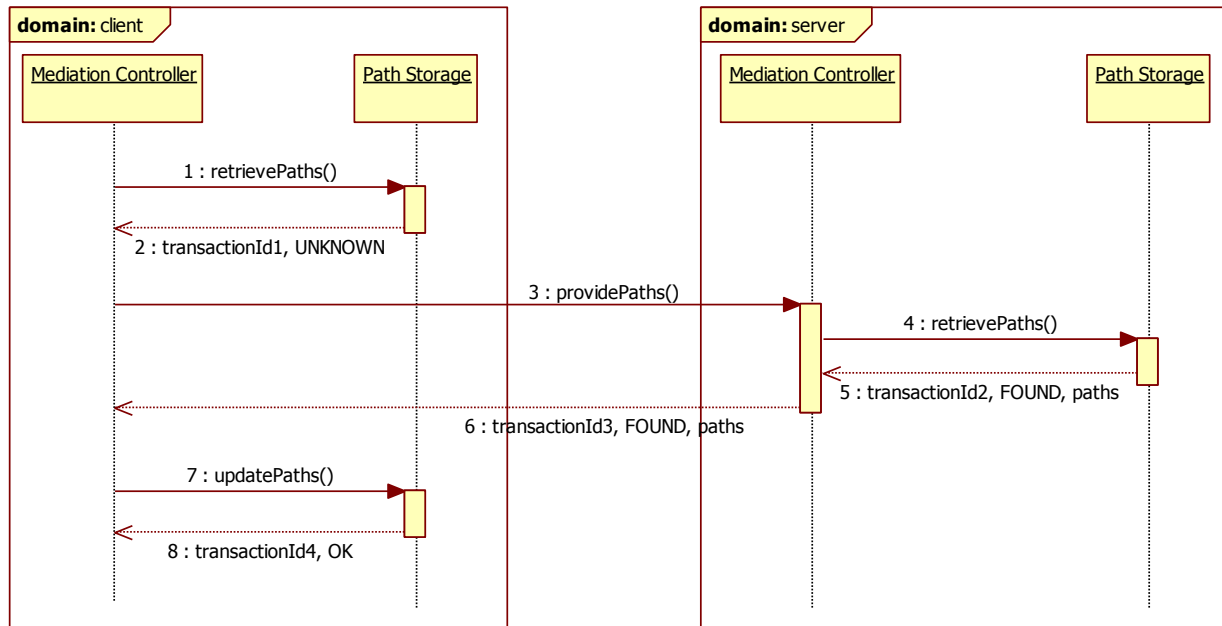


Figure 3-13: Path Discovery sequence diagram for paths not available in local cache

Description of the messages:

1. retrievePaths(transactionId, clientId, serverIp, classOfService)  
Path Storage performs longest prefix matching against the Type 3 data for given *classOfService*.
2. transactionId1, UNKNOWN, paths  
In this diagram we assume that result of matching is unknown (status=UNKNOWN).
3. providePaths(transactionId3, clientId, serverIp, classOfService)  
Mediation Controller forwards the request to the server side Mediation Controller. Before sending this message, it starts a timeout timer for *transactionId3*. The expiration of timer is equal to the result NOTFOUND (there are no paths supporting this pair of IP addresses).
4. retrievePaths(transactionId2, clientId, serverIp, classOfService)  
Server side Mediation Controller asks its own Path Storage for paths.
5. transactionId2, FOUND, paths  
In this diagram we assume that matching is successful (status=FOUND) and paths are returned. However, if there are no paths between serverIp and clientId addresses, the result NOTFOUND should be returned. Paths have the following structure:
  - a list of path information structures, where each structure has:
    - AS path as a sequence of integer values,
    - Source prefix is related to serverIP address
    - Destination prefix is related to clientIP address
    - QoS parameters: IPLR, IPTD, maximum BW for a single delivery. These parameters should correspond to end-to-end (aggregated) values (Type 3). Therefore, Path storage component at server side should calculate them based on Type 1 and Type 2 information received from RAE. The preparation of Type 3 information involves calculation of path parameters from Type 1 and Type 2 information (Type

3=Type1+Type2). Note that IPTD and IPLR parameters are additive, while maximum BW is calculated as minimum of arguments.

6. transactionId3, FOUND, paths

Paths are forwarded to the client side domain. At this moment the Path Discovery is completed, however two additional messages are exchanged in parallel.

7. updatePaths(transactionId4, classOfService, paths)

Mediation Controller updates the information in the Path Storage, where path caching is performed. For each path stored in cache, we start timeout. The value of timeout should be configured by CME administration interface or provided by CME configuration file. Basically, timeout should be related to life time of paths.

8. transactionId4, OK

Path Storage confirms the update of the cache.

### 3.4.4 Server Awareness

Server Awareness is the operation a CME learns of the status of the servers distributing the content requested by a user. It consists of two different operations:

- The CME interrogates a remote CME about the status of the candidate servers located in the remote ISP.
- The remote ISP (or ISPs) obtains this information from their associated SNMEs.

Both operations are explained below:

#### 3.4.4.1 Inter CME

After successful name resolution, the Mediation Controller of the client-side CME will request the load for all content servers, included in the first content source (from the sorted content sources' list), from their respective server-side CMEs. More specifically, client-side CME will gather all content servers adjacent to the same CMEs and it will include multiple ServerLoadRequests in its message sent to each server-side CME, while waiting for the ServerLoadResponses, containing servers' load. The procedure is presented in sequence diagram below.

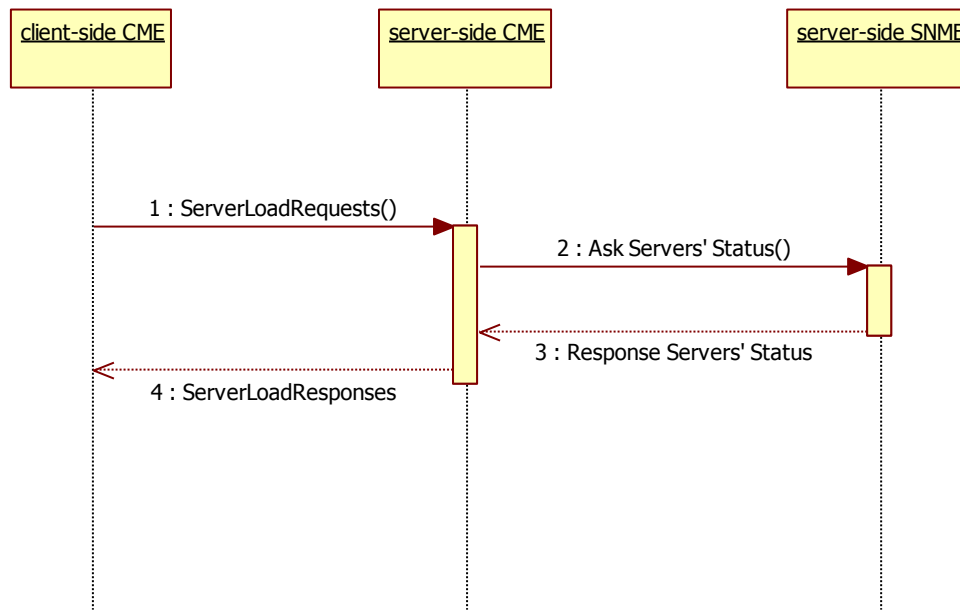


Figure 3-14: inter-CME communication in Server Awareness

The server awareness process is supported from the inter-CME protocol, with the inclusion of the following messages:

- ServerLoadRequest:
  - Content Server IP address
- ServerLoadResponse:
  - Content Server IP address
  - Server Status
    - 1 for LOW
    - 2 for MEDIUM
    - 3 for HIGH
    - 4 for DOWN

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]



### 3.4.5 Decision Process

The decision process selects the best duple [server, path], which will be used to deliver content requested by the client. This process is performed by Decision Maker once the Mediation Controller gathers information about the content characteristics and its location (from Content Record), content servers' load (from Servers Awareness Mediator) and available paths (from Path Storage). If given content is published with several content sources that differ in content coding, bit rate, CoS, priority, etc., the Mediation Controller creates the list of content sources that match CoSs subscribed by user with content source priority and supported CoSs. This list should be ordered based on the following criteria:

- No sources with CoS higher than Client's CoS should be considered (i.e. if Client's CoS is Best Effort, Premium and Better Than Best Efforts sources should be directly discarded).
- For a Client's CoS, those sources matching its CoS should be placed in the first position of the list, and then those with lower CoS (i.e Premium → Better than Best Effort → Best Effort)
- For the same CoS, the sources should be ordered according to priority.

Next, Mediation Controller chooses the first content source in the ordered list and gathers required information (paths, loads, etc.) for all servers in selected content source. Finally, it invokes Decision Maker to select preferred server and path.

In case of a negative response, i.e., when the Decision Maker returns a NOTFOUND message, the next source within the sorted list should be considered.

Figure 3-19 presents the message sequence diagram for the Decision Maker.

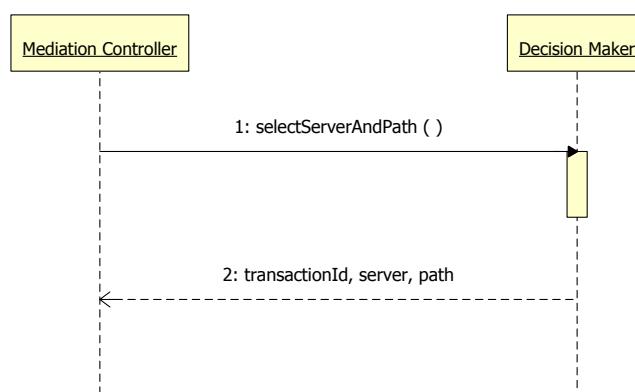


Figure 3-19: Decision Maker message sequence diagram

### Description of the messages:

1. selectServerAndPath (transactionId, classOfService, trafficDescriptor, qosConstraints, serversAndPaths)

The parameters are the following:

- transactionId – identifies the message exchange
- classOfService – identifier of the COMET CoS (coming from Content Record),
- trafficDescriptor – a single value describing traffic profile Bit Rate (*BR*). This values comes from Content Record,
- qosConstraints – set of parameters describing QoS constraints, i.e., IPTD, IPLR, obtained from Content Record,
- serversAndPaths, the list of information structures that describe for each server and list of paths going from the server to the client. Each structure contains:
  - serverIP – IP address as an identifier of the server (coming from Content Record),
  - serverLoad–load of the server (coming from SNME). It is one of the following values LOW, MEDIUM, HIGH or UNKNOWN,
  - List of paths:
    - AS path as a sequence of AS numbers (coming from Path Discovery),
    - QoS parameters: IPTD, IPLR and maximum BW for a single delivery (coming from Path Discovery)
    - Optional: path load measured by client's domain over the path (coming from SNME).

2. transactionId, FOUND, server, path

In positive scenario, the response includes serverIP and ASpath, while in negative case the response is NOTFOUND.

### Details of the decision algorithm:

The complex set of parameters used by Decision Maker requires a multi-criteria decision algorithm. These algorithms are widely studied in the Multiple Criteria Decision Analysis (MCDA) [19],[20]. MCDA splits into single and multiple objective optimization. In general, the Decision Maker requires Multi-objective optimization [21]. The basic model of Multi-objective optimization defines the decision space  $\mathcal{R}$ , which consists of the candidates (decision vectors)  $x=(x_1, x_2, \dots, x_i)$ . Each decision vector contains 5 decision variables related to server load and path characteristics (path length, packet delay, packet losses, and bandwidth). Each decision variable has assigned attribute, which determines whether selected solution must meet constraints (strict mode) or may exceed constraints (tolerant mode). In order to evaluate the impact of particular decision variable, decision algorithm uses two reference parameters, called reservation level and aspiration level. The reservation level is the upper limit for decision variable, which should not be crossed by preferred solution (in strict mode it cannot exceed the reservation level). On the other hand, the aspiration level defines the lower bound for decision variable, beyond which preference of evaluated solutions is similar. The operator of COMET system may influence on behavior of decision algorithm by appropriate setting of reservation and aspiration levels as well as the strict or tolerant mode. In chapter 6, we present results of our studies related to evaluation of 4 decision strategies. Those strategies cover: (1) random selection of server and delivery of content using shortest path (strategy used in the current Internet), (2) selection of closest server and delivery of content using shortest path (strategy of some CDNs), (3) selection of the least loaded server and delivery of content using shortest path (strategy of some CDNs), (4) selection of the least loaded server and delivery of content using the least loaded path which may not necessarily be the shortest path (strategy assumed in COMET).

The decision algorithm consists of the following steps:

1. Decision Maker creates decision space based on the received information as the list of candidate solutions. Each candidate is a structure of decision variables, which has the following form:

Candidate [serverIP, ASpath]

```
{
  serverLoad – this is numerical representation of server status. The following
                mapping is applied LOW=1, MEDIUM=2, HIGH=3, UNKNOWN=3.1,
  pathLength – the path length denotes the number of ASs on the path. It is
                calculated based on the AS path parameter,
  IPTD – IPTD of the path,
  IPLR – IPLR of the path,
  BW – maximum supported bandwidth for a single content delivery over path,
}
```

2. Decision space should include at least one candidate, otherwise return NOTFOUND. When the list of candidates is very large, limit the number of considered candidates to some reasonable value, e.g. Max\_number\_of\_candidates=1000.
3. If at least one decision variable is configured in strict mode, then Decision Maker should remove unfeasible candidates from decision space. We treat a candidate as unfeasible if at least one decision variable that were configured as strict has value greater than reservation level. After this step, decision space should include at least one candidate, otherwise return NOTFOUND.
4. Decision Maker calculates the rank value  $R_i(.)$  for each candidate. It uses objective function with reservation and aspiration levels specific for each decision variable.

$$R_i(.) = \min_{k=1,...,5} \left[ \frac{r_k - q_k}{r_k - a_k} \right]$$

where:  $i$  is the number of candidate,  $k$  is the number 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 fraction of reservation level by aspiration coefficient  $\alpha_k$ ,  $a_k = \alpha_k \times r_k$ . In Table 3, we present recommended values of reservation and aspiration levels for considered decision variables:

Decision variable ( $q_k$ )	Reservation level ( $r_k$ )	Aspiration coefficient ( $\alpha_k$ )
$q_1$ =server Load	$r_1=3$	$\alpha_1=1/3$ *
$q_2$ =path Length	$r_2=10$ *	$\alpha_2=1/5$ *
$q_3$ =IPTD of the path	$r_3$ = IPTD from QoS constraints	$\alpha_3=2/3$ *
$q_4$ =IPLR of the path	$r_4$ =IPLR from QoS constraints	$\alpha_4=2/3$ *
$q_5$ =BR from traffic descriptor	$r_5$ = BW of the path	$\alpha_5=1.0$

\* values of these parameters should be configured by CME administration interface. The domain operator may influence the importance of particular decision variables by tuning values of aspiration coefficients. The maximum importance is for  $\alpha_k = 0$ , while for  $\alpha_k = 1$ , the decision variable has no influence.

Table 3: Recommended values of reservation and aspiration levels

5. Select the candidate with maximum rank as the best candidate and provide response FOUND (serverIP, ASpath). In case of multiple candidates have the same rank, randomly select one of them.

### 3.4.6 Path Configuration

As the outcome of the Decision Process, the Mediation Controller knows IP address of server and selected path (in the form of list of AS numbers) that should be used for content consumption. In next step, the Path Configurator prepares forwarding plane for content delivery from server's IP address to client's IP address. The Path Configurator performs two basic actions:

1. it translates the selected path, which has the form of "list of AS numbers", into the list of forwarding rule indicators (keys). These keys are used by consecutive CAFE to forward packets.
2. it configures the CAFE adjacent to the content server to intercept IP packets and encapsulate them with COMET header, which includes list of forwarding rule indicators (keys).

In this chapter we briefly present the path configuration process. The detailed specification is included in section 2 of Annex B. Note that this specification focuses only on CME level interactions, therefore the following elements are out of the scope of this document:

- specification of CAFE forwarding is included in D4.2 [1]
- specification of CAFE configuration agent, is also included in D4.2 [1];

#### 3.4.6.1 Information required for path configuration

The Path Configurator requires information about CAFEs located inside own domain as well as border CAFEs located in peering domains. This information is stored in the following tables.

**Table 1** allows Path Configurator to map the local IP address of client or server into the CAFE handling the traffic passing from/to this address. Formally, we define T1 as:  $(IP\ address) \rightarrow (CAFE\ id)$ . The exemplary Table 1 is show below:

IP prefix	CAFE id
2.2.2.0/24	1.16
194.29.0.0/16	1.17

**Table 2** allows Path Configurator to translate peering ASs from list of AS numbers into pair of CAFEs: one belonging to its domain and one belonging to peering domain. Moreover, this table is used to determine the IP address/port of CME in peering domain. Formally, we define T2 as:  $T2(\{AS\ path\}, CoS) \rightarrow (source\ CAFE\ id, sink\ CAFE\ id, peering\ CME\ IP\ address)$ . The exemplary Table 2 is show below:

AS peering	CoS	Source CAFE id	Sink CAFE id	Peering CME
1,2	BE	1.16	2.5	2.2.2.2:9876
2,3	PR	1.16	3.1	3.3.3.3:8765

**Table 3** allows Path Configurator to translate the sequence of CAFEs into the forwarding keys, which are used by CAFEs to determine where transfer packet. Formally, we define T3 as:  $T3(\{CAFE\ id_1, ..., CAFE\ id_n\}, CoS) \rightarrow (forwarding\ key)$ . Note that we can distinguish two types of

sequences of CAFE ids: 1) peering to next domain, and 2) transit across current domain. The exemplary Table 3 is show below:

CAFE peering	CoS	Forwarding key
{1.16, 2.5}	BE	{0xa1, 0xbb}
{1.16, 3.3}	PR	{0x0f}

When Path Configurator configures new path, it stores this information in **Table 4**. This table includes a set of active paths going from server's domain to client's domain. This table is updated by COMET system. Formally, we define T4 as:  $T4(\{AS\ path\}, CoS) \rightarrow (forwarding\ key, CAFE\ id)$ . Details of this mapping are following:

AS path	CoS	Sequence of forwarding keys	egres CAFE id
{1,2,3}	BE	{0xaa, 0x01, 0x17}	1.16
{1,2}	PR	{0xaa, 0x05}	1.16

### 3.4.6.2 Message sequence diagrams

Figure 3-20 presents the message sequence diagram for Path Configuration. This diagram also covers Path Provisioning sub-process which must be performed when requested path has not been yet configured and therefore cannot be mapped with T4.

The Path Configuration starts at client's domain after finishing Decision Process when the following information is known:

- AS\_PATH – AS path from server to the client as a list of AS numbers,
- IPC – IP address of the client,
- IPS – IP address of the server,
- FILTER – transport protocol and port numbers (at least one port must be known to uniquely identify flow ),
- IPCME – IP address of the server side CME (from content record),
- BR – requested bit rate (from content record),
- CoS – requested Class of Service (from content record).

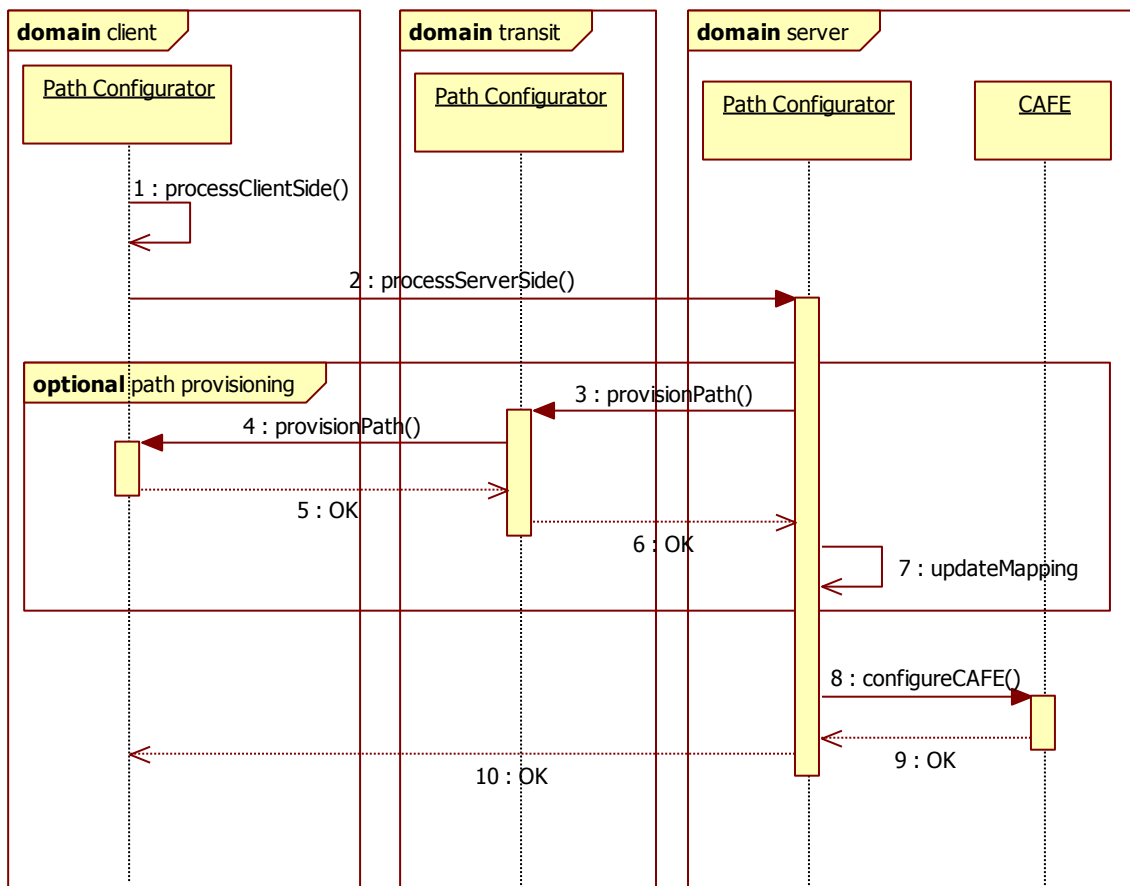


Figure 3-20: Path Configuration message sequence diagram

## Description of the messages:

## 1. processClientSide(AS\_PATH, IPC, IPS, FILTER, BR, CoS)

Client side process finds key used for packet forwarding from ingress CAFE of client's domain to the client. It performs the following steps: (1) it finds CAFE id where the client is connected (using Table 1), (2) it finds ingress CAFE where traffic enters the domain (using Table 2), and (3) it finds forwarding *key\_client*, which allow to transfer packets from ingress CAFE to client's CAFE (using Table 3). Moreover, if client domain requires any resource control functions or other specific configuration actions they are performed before going to the next step

## 2. processServerSide(AS\_PATH, IPC, IPS, FILTER, BR, CoS, KEY)

The objective of server side process is to find key used for packet forwarding from server IP address towards the ingress CAFE of client's domain. In this process we distinguish two sub-processes. First, called path provisioning, is responsible for finding *key\_path* used for packet forwarding from the egress CAFE of server's domain to ingress CAFE of client's domain. This process is invoked only for the first new request along given path. For next requests, we reuse *key\_path* cached in Table 4. Note that *key\_path* could also be statically configured following the prearranged agreements between ISPs. In this case, we use CME administration interface to put information to Table 4. The second sub-process is responsible for finding *key\_server* used for packet forwarding from the server CAFE to the egress CAFE of server's domain.

In the diagram we assumed that *key\_path* has not been yet provisioned, therefore path configurator invokes optional path provisioning sub-process.

\*\*\* *Optional Path Provisioning Procedure starts* \*\*\*

3. provisionPath() in transit domains

This message is transferred between CME along the selected path towards the client's domains to translate the "list of AS numbers" into the sequence of "forwarding keys". Each transit domain finds ingress and egress CAFEs using Table 2 and updates the list of forwarding keys based on Table 3. Moreover, if transit domain requires any resource control functions they are performed before going to the next step

4. provisionPath() in client's domain:

Once this message reaches the client's domain, the path configurator creates the *key\_path*.

5. 6. OK ()

The OK message is send back to server's domain and confirms that path is correctly provisioned. It also carries *key\_path*.

7. updateMapping()

The *key\_path* is stored in Table 4.

\*\*\* *Optional Path Provisioning finishes* \*\*\*

Now, the path provisioning sub-process is finished, so server side processing is continued. It performs the following steps: (1) it finds CAFE id where the server is connected (based on Table 1), (2) it finds forwarding *key\_server*, which allows to transfer packets from server CAFE to egress CAFE and (3) creates the final forwarding key as concatenation of *key\_client* received from client side, *key\_path* and *key\_server* created at server side. Moreover, if server domain requires any resource control functions they are performed before going to the next step

8. configureCAFE(IPS, IPC, FILTER, BR, AS\_PATH, KEY, REFRESH\_TIME)

This message configures CAFE at the server side (with the aid of CAFE Configurator module). The CAFE intercepts packets based on configured classifier and encapsulates them with COMET header, which includes the forwarding key.

9. 10 OK()

At this moment CAFE is configured. In this way, the path selected in decision process is configured to delivered content from server to client. Therefore, the Path Configurator at client side informs Mediation Controller that path configuration has finished.

### 3.4.7 CAFE Configuration

The CAFE Configurator is responsible for configuring CAFEs located at servers' or client's side. It also collects information about terminated flows and keeps track whether CAFEs are alive. The CAFE Configurator uses the following messages:

1. configureStream()

This message is used to configure the edge CAFE to intercept IP packets. It includes the following fields:

- flow id – integer identifying the configured flow,
- filter – set of fields for capturing IP packet streams. The filter covers: IP source address, IP destination address, protocol, source port number and destination port number.
- bandwidth – integer specifying amount of resources assigned to the flow (in kbit/s),
- cos – class of service used for handling the stream,
- forwarding key – a vector of bytes used for packet forwarding,
- refresh\_time – integer indicating timeout for detecting end of the flow,
- as\_path – sequence of integers indicating the inter-domain path.

The response is one of the following messages {flow id, CAFE\_SUCCESS} or {flow id, CAFE\_FAILURE}.

## 2. collectExpiredStreams()

This message is sent periodically by CAFE Configurator to get information about terminated flows. The CAFE responses with the following entries:

- flow id – integer identifying the stream,
- bandwidth – integer specifying amount of resources assigned to the stream (in kbit/s),
- cos – class used for handling the stream,
- as\_path – sequence of integers indicating the inter-domain path,
- transferred\_bytes – amount of data transferred,
- flow\_duration – in seconds.

The information about terminated flows is provided to Path Configurator and SNME module (optionally) in order to release resources and update statistics.



## 4 Coupled Approach Specifications

### 4.1 Overview

In this chapter, we present the coupled approach, an Internet-scale content access and delivery platform, in order to enable native “in-network” content diffusion services with active involvement from network operators. The global diffusion of content across the Internet is achieved through collaborative content resolution and delivery functions between individual ISP networks. With such an approach, content providers/owners and consumers may publish or consume content through issuing a set of unified content manipulation primitives only to their local ISPs. This feature exhibits salient differences from existing overlay-based approaches where a content provider or consumer may need to interact with multiple content overlays (in terms of content registration/publication or request) in order to maximize the accessibility of the content to be shared, or the opportunity to find the content requested.

The coupled approach aims to enable a future content-centric Internet that will overcome the current intrinsic constraints by efficiently diffusing media content of massive scale. It entails a holistic approach, supporting content manipulation capabilities that encompass the entire content life cycle, from content publication to content resolution and, finally, to content delivery. It provides to both content providers and consumers high flexibility in expressing their location preferences when publishing and requesting content, respectively, thanks to the proposed scoping and filtering functions. Content manipulation operations can be driven by a variety of factors, including business relationships between ISPs, local ISP policies, and specific content provider and customer preferences. Content resolution is also natively coupled with optimized content routing techniques that enable efficient unicast and multicast-based content delivery across the global Internet.

The original Internet model focused mainly on connecting machines whereby addresses point to physical end-hosts and routing protocols compute routes to specific destination endpoints. Nowadays the Internet is primarily used for transporting content/media, where a high volume of both user-generated and professional digital content, e.g., webpages, movies/songs, live video streams, etc., is delivered to users who are usually only interested in the content itself rather than the location of the content sources. Human needs along with the nature of communication technologies have transformed the Internet into a new content marketplace, generating revenue for various stakeholders. In fact, the Internet is rapidly becoming a super-highway for massive digital content dissemination.

In this context, many researchers have advocated a transition of the Internet model from *host-centric* to *content-centric*, with various architectural approaches proposed [1][8][7][9][10][11][12]. Many of these proposals support the key feature of *location independence*, where content consumers do not obtain explicit location information (e.g., the IP address) of the targeted content source *a priori*, before issuing the consumption request [1][8][7][10][12]. Nevertheless, location requirements are still demanded by both content consumers and providers. On the one hand, content providers may want their content accessed only by content consumers from a specific region (known as *geo-blocking*), e.g., BBC iPlayer, Amazon Video-on-Demand, Apple iTunes Store and Sina video services. On the other hand, content consumers may prefer content originated from specific regions in the Internet, for instance, a US-based shopper might only like to check the price of an item sold in Amazon online stores in North America rather than anywhere else in the world. Today, this is typically achieved through the user’s explicit input in the URL (e.g., Amazon.com and Amazon.ca), and supported by name resolution through the standard Domain Name System (DNS) [13], with the relevant requests directed towards the specific regional web server. Similar practice can be observed in multimedia-based content access (e.g., video-on-demand services), where consumers have specific requirements regarding the location/area of content sources.

In this chapter, we detail the coupled approach. The objective is to both accurately and efficiently “hit” (or “not hit”) content objects in *specific regions/areas of the Internet*, based on user requirements and preferences. Such an approach, deployed by ISPs, allows both content providers

and consumers to express their location requirements when publishing/requesting content, thanks to the supported content *scoping/filtering* functions. In particular, instead of following the conventional DNS-like approach, where a content URL is translated into an explicit IP address pointing to the targeted content server, the proposed content resolution scheme is based on hop-by-hop “gossip”-like communication between dedicated Content Resolution and Mediation Entity (CRME) entities residing in individual ISP networks. Content resolution operations can be driven by a variety of factors, including the business relationships among ISPs (provider/customer/peer), content consumer preferences and local ISP policies. This resolution approach is natively coupled with content delivery processes (e.g., path setup), supporting both unicast and multicast functions. Specifically, a content consumer simply issues a single *content consumption request* message (capable of carrying his/her location preferences on the content source candidate(s)), and then individual CRME entities collaboratively resolve the content identifier in the request, in a *hop-by-hop manner*, towards the desired source. Upon receiving the request, the selected content source starts transmitting the requested content to the consumer. During this content resolution operation, “multicast-like” content states are installed along the resolution path so that the content flows back immediately upon the completion of the resolution process. By exploiting multicast delivery techniques, we increase the sustainability of the system in view of the expected explosion of content in the Internet.

## 4.2 Basics

### 4.2.1 Entities

In the coupled approach, content manipulation operations rely on two distinct entities that encompass the functionalities defined in the COMET architecture. These entities are:

- Content Resolution and Mediation Entity (CRME) at the Content Mediation Plane (CMP)
- Content-aware Forwarding Entity (CAFE) at the Content Forwarding Plane (CFP)

#### 4.2.1.1 Content Resolution and Mediation Entity (CRME)

A CRME primarily handles content publication requests, discovers the requested content and supports the delivery of content while the CAFE collaborates with its local CRME(s) to enforce receiver driven content delivery paths.

The CRME is envisioned to be present in each domain for (1) handling *local* publication requests and content consumption requests and (2) for interacting with their neighbouring counterparts for content publication/resolution across domains. Both content servers and clients are configured to know their local CRME(s). The number of CRME within each domain depends on performance and resilience considerations. For example, having more CRME(s) within a domain will provide higher redundancy and thus increase resilience in case of CRME failures.

Figure 4-1 depicts the functional view of the overall infrastructure; we explain the operational properties of each functional block below. The internal structure of the CRME consists of four logical components. It is responsible for dealing with requests from both content providers and consumers (via CRME-CS and CRME-CC interfaces respectively).

The *Content Resolution Function* (CRF) acts as a content record repository which contains a database of content records that have been propagated or registered within a domain. The CRF allocates content IDs and creates the new content record upon reception of a new content registration instruction. The specific rules on how the content record is created and disseminated are explained in the next sections.

The *Content Mediation Function* (CMF) coordinates and mediates the various content-related operations. It also holds the responsibility of content ID lookup upon each content consumption request from a content client. When a content consumer via the content client sends a content request, the request first reaches the CMF which triggers a communication between CMF and CRF

to resolve the request. The CMF will contact its local CRF to verify if the requested content exists within the local index before proceeding to determine the next resolution step.

The *Path Management Function* (PMF) maintains a record of ingress and egress Content-aware Forwarding Entity (CAFE) within the local domain for each active content session being delivered in the network. CMF communicates with PMF for path discovery purposes whereby PMF collects offline routing information from the network (e.g. BGP reachability information) and then enables the CMF to find the possible path(s) for the delivery of a content. To compute the best path, CMF uses additional monitoring information gathered from the *Server and Network Monitoring Function* (SNMF). SNMF gathers necessary “near real-time” information on content server and underlying network conditions for supporting optimized content resolution and delivery configuration operations. These information are then fed to the CMF to enable the mediation of content delivery path.

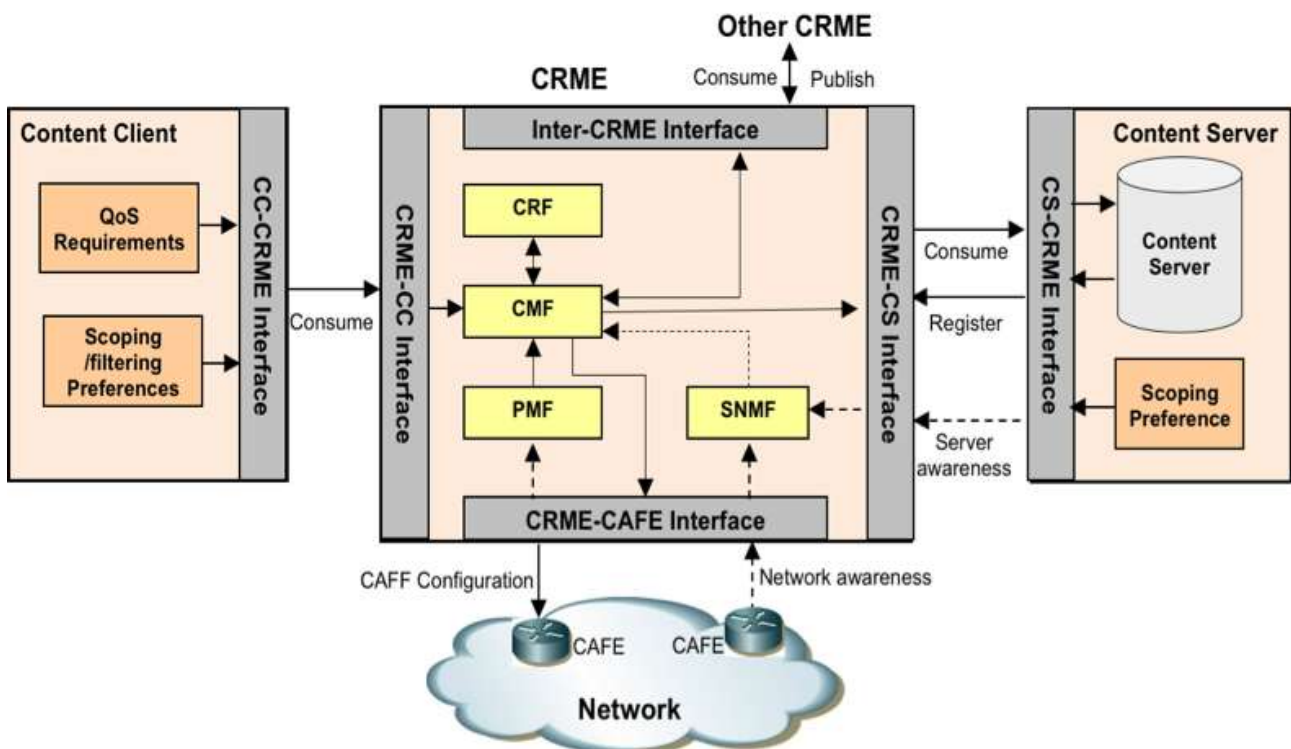


Figure 4-1: High-level functional view of the hop-by-hop hierarchical content resolution approach

#### 4.2.1.2 Content-aware Forwarding Entity (CAFE)

A CAFE is a network element that is able to natively process content packets according to their IDs. It is considered as residing in the CFP. In general, it is not necessary for every router in the network to be a CAFE, and typically CAFEs are planted at the network boundary as ingress and egress points for content delivery across ISP networks. It is responsible to maintain states for content delivery. It interacts with its local CRME to construct the delivery path during the content resolution process.

The detail function of CAFEs will be specified in D4.2 with the description of content delivery process.

### 4.2.2 Interactions

CRMEs communicate with other entities via specialized interfaces as described below:

- Inter-CRME interface – enables interaction amongst CRMEs in neighbouring domains especially when they cooperate in content publication and searching for a requested content across domains.

- CRME-CS interface – connects content servers owned by content providers with CRMEs, and allows content providers to publish content, optionally with scoping (see next section) requirements on potential content consumers. This interface is also responsible for passing information on server load conditions to a CRME for enabling optimized content resolution operations.
- CRME-CC interface – connects content clients with the CRMEs and allows consumers requesting and receiving content with scoping/filtering preferences on candidate content sources.
- CRME-CAFE interface – bi-directional interface to allow a CRME to actively configure relevant CAFEs for each content session (e.g. content state maintenance). It also gathers necessary information from the underlying network that will be used for optimized content resolution processes.

### 4.2.3 Provider Route Forwarding Rule

We define the “*provider router forwarding rule*” which will be the central lynchpin of the coupled approach on which the content publication and resolution processes will be based. The rule is simple yet effective in disseminating the necessary content information to the necessary locations so that a content resolution process running based on the same rule can find the requested content. The rule states that the information (e.g., the `Consume` message) should be passed along the provider route only. If more than one provider links are found, then a random one is chosen. We provide two examples in Figure 4-2.

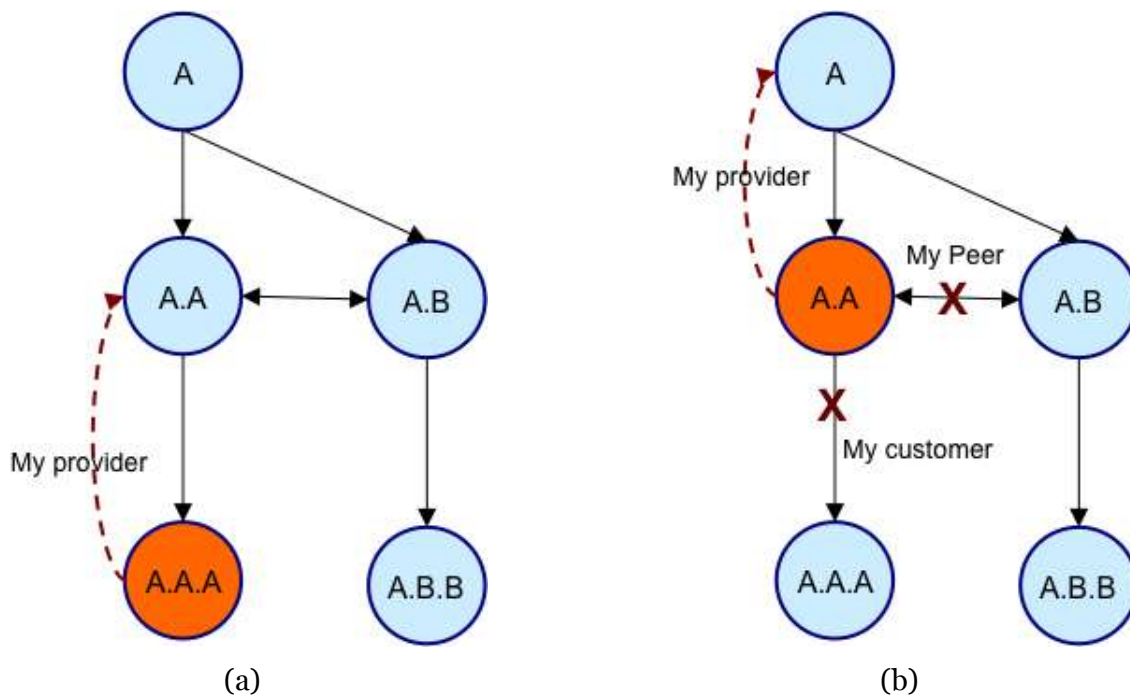


Figure 4-2: Provider route forwarding rule; (a) from domain A.A.A's perspective, (b) from domain A.A's perspective

Figure 4-2 depicts a simple *domain-level* network topology with each circle logically representing a domain containing a CRME. The double-headed arrows indicate peering relationship between the two connected domains while the single-headed arrows depict provider-customer relationship with the direction from provider towards customer.

Following the *provider route forwarding rule*, domain A.A.A has A.A as its provider and no other relationship. Thus, according to the rule, A.A.A will only forward the message it has to A.A. On the other hand, domain A.A has three inter-domain links – (a) domain A as its provider, (b) domain A.B as its peer and (c) domain A.A.A as its customer. Following the rule, A.A will only forward the message it has to domain A and not to its domain A.B and A.A.A. Note that this rule naturally

forms a valley-free forwarding path (e.g., domain **A.A** will not forward its message down and back to domain **A.A.A**).

The content operation involves the following three-stage lifecycle: *publication*, *resolution* and *delivery*. The task of content resolution is to (1) identify the desired content source in the Internet according to the requested content ID and optionally content consumer preferences, and (2) actually trigger the content transmission by the selected content server. Once the content server starts the transmission of the content upon receiving the content consumption request, the content delivery function is responsible for enforcing the actual delivery path back to the consumer.

### 4.3 Content Publication

Content publication is the process of making content available across the Internet. It consists of two stages.

*Stage 1: Content Registration* – It begins with the content provider notifying the local CRME that a new content is now available via a `Register` message. In the case where multiple copies of the same content are available at different locations, the content provider is responsible for informing the local CRME of each content server hosting that specific content copy. Upon reception of the `Register` message, the CRME registers this content by creating a new record entry in its local content management repository containing (1) a globally unique content ID assigned to that content, and (2) the *explicit location* of the content (i.e. IP address of the content server).

*Stage 2: Content Publication Dissemination* – Once the content is registered to a CRME, this CRME is responsible for publishing it globally to ensure successful discovery by potential consumers. This is achieved through the dissemination of the `Publish` message across CRME(s) in individual domains according to their business relationships. A `Publish` message is created by the CRME where the content is actually registered by the content provider. By default, each CRME disseminates a new `Publish` message towards its counterpart in the *provider domain(s)* until it reaches a tier-1 ISP network. Each CRME receiving a new `Publish` message updates its content management repository with a new record entry containing the content ID and the *implicit location* of the content (i.e. the IP prefix associated with the neighbouring domain from where the `Publish` message has been forwarded). Following this rule, each CRME effectively knows the locations of all the content within its own domain (explicitly) and those under it (its customer domains, implicitly). Peer domains, however, will not know the content records of each other.

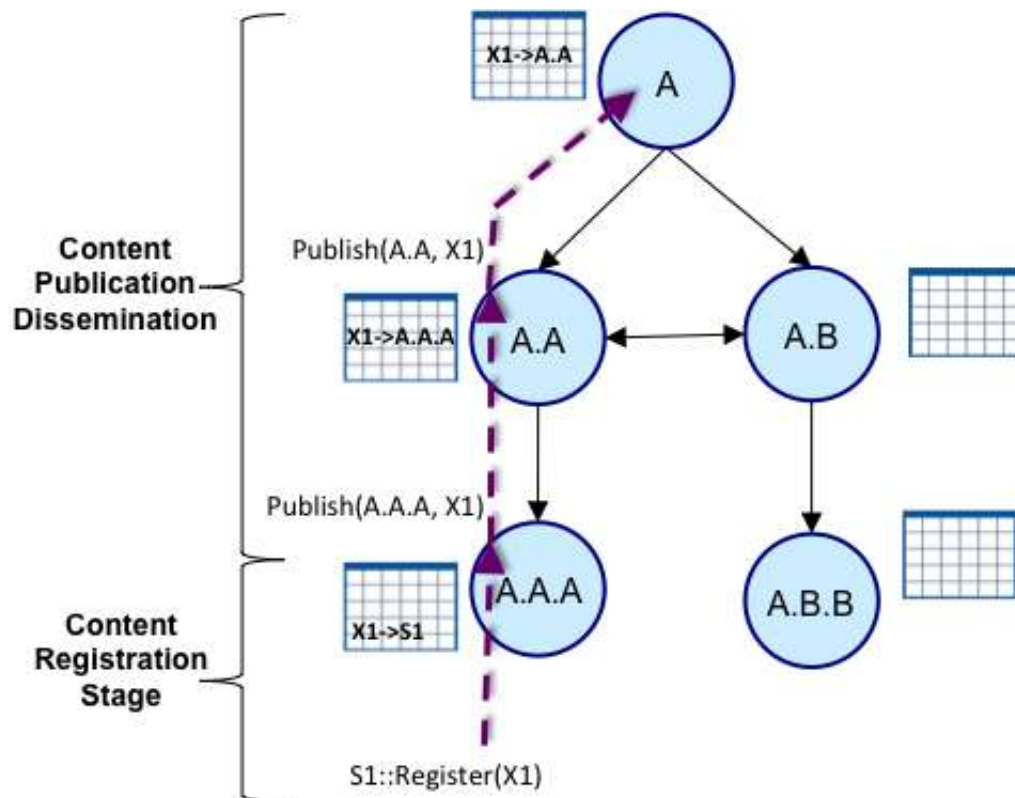


Figure 4-3: Content publication based on provider route forwarding rule.

We illustrate in Figure 4-3 the simplest case of content publication using the provider route forwarding rule explained in section 4.2.3. When **A.A.A** receives the `Register` message from **S1**, it registers content **X1** by creating a new record in its content management repository (stage 1) and then proceeds to send a `Publish` message to its provider (in this case, domain **A.A**, stage 2). Abiding to the same provider route forwarding rule, domain **A.A** continues to disseminate the `Publish` message to tier-1 domain **A**, which signifies the end of the publication process for content **X1**. We illustrate the above in UML representation in Figure 4-4 whereby the process stops at the tier-1 domain.

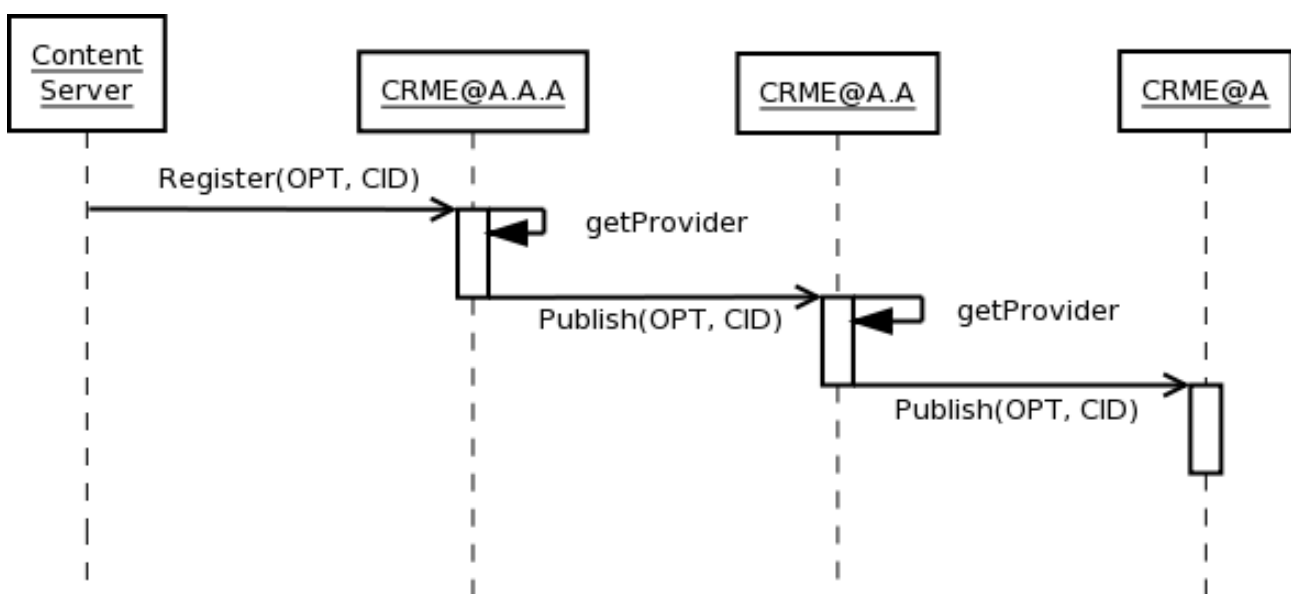


Figure 4-4: UML representation of the content publication process



We introduce the concept of *scoped publication* to allow publication of content only to specific areas in the Internet as designated by the content provider. This feature is able to natively support regionally-restricted content broadcasting services such as BBC iPlayer and Amazon VoD that are only available within the UK and the US respectively. We achieve this through the INCLUDE option embedded in the Register/Publish messages where the content provider specifies a scoped area in the Internet, e.g. only the IP prefix associated with the local ISP network where the content is registered. A special case of *scoped publication* is the *wildcard mode* (denoted by asterisk “\*” symbolizing *all* domains) for which the content provider has no restrictions on the geographical location of potential consumers in the Internet.

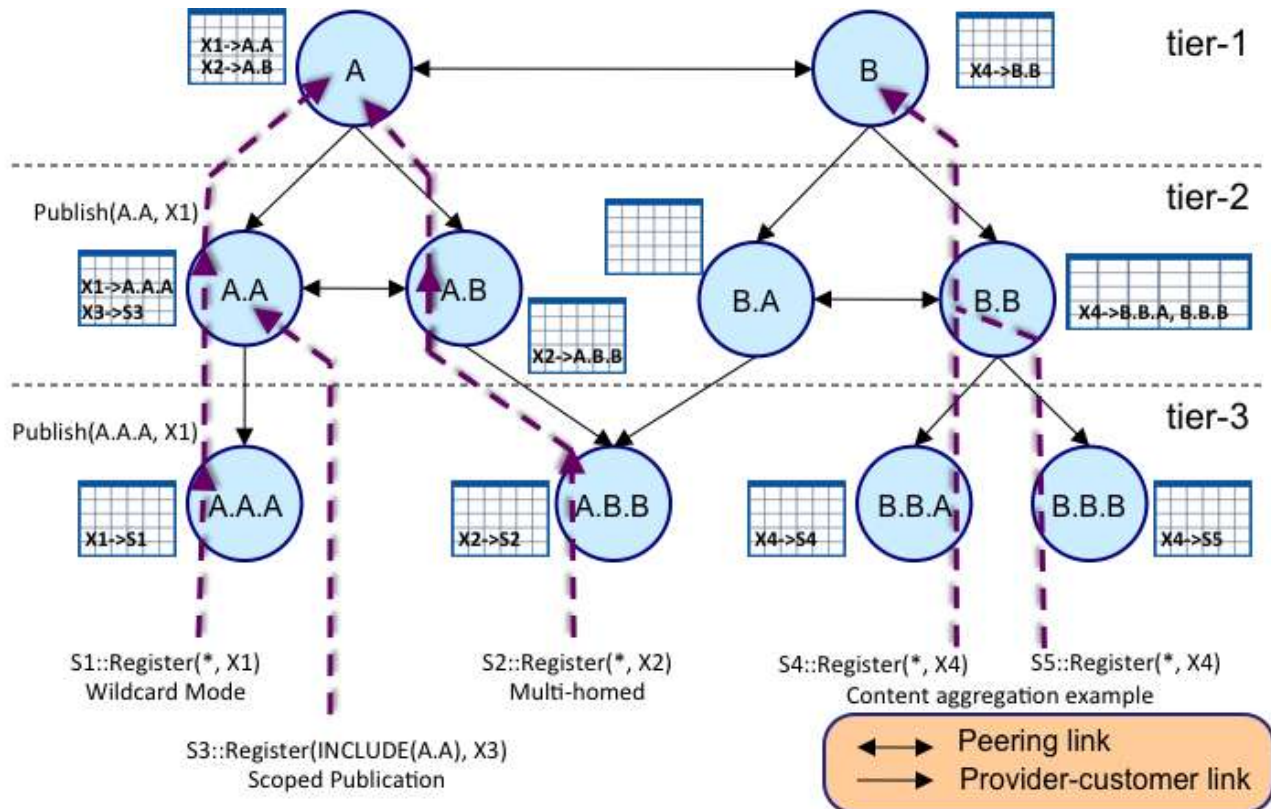


Figure 4-5: Content publication in different modes

Figure 4-5 illustrates different scenarios invoking the different modes in the publication process.

- *Wildcard mode* – Assume that content provider **S1** registers a content item (assigned with ID **X1** by the local CRF in the stub domain **A.A.A**) to the entire Internet by issuing a Register message with a wildcard. Each intermediate CRF along the publication path creates a content entry for **X1** associated with the IP prefix of its customer domain from where the Publish message has been forwarded. For clarity, the Publish messages are omitted in subsequent figures in this section.
- *Multi-homed scenario* – In cases where a domain is multi-homed (e.g., domain **A.B.B** in our example), the forwarding of the Publish messages can follow several options. First, the CRME can broadcast the Publish message to all its providers. Second, the CRME can choose one provider to which to forward the Publish message. This can be dictated by local domain policy. Finally, if no policy exists to influence the choice of the provider, then the CRME can choose one randomly. Note that local policies can be used to influence the forwarding of messages in general and not restricted to the multi-homed case. In the figure, the first option is illustrated whereby the assumption is that **A.B.B** prefers **A.B** than **B.A**. The second option will result in a concurrent Publish message to **B.A** and **B**.

- *Scoped mode* – **S3** illustrates the *scoped* registration by only registering content **X3** to tier-2 domain **A.A** from this content provider. This effectively limits the access of content **X3** to domain **A.A** and its customer domain **A.A.A**.
- *Aggregation example* – Finally, records for different copies of the same content can also be aggregated in content publication processes. In the figure, both **S4** and **S5** host one copy of content **X4** respectively, but the two content publication messages from **B.B.A** and **B.B.B** are merged at **B.B**, in which case domain **B** only holds one record with an aggregated location information (**X4**→**B.B**). Upon the actual content consumption request for **X4** from a content client, **B.B** can forward it to either **B.B.A** or **B.B.B** based on performance conditions gathered from SNMF such as content delivery path quality or server load, as will be discussed later.

## 4.4 Content Resolution

In the content resolution process, a content consumption request issued by a content client is resolved by discovering the location of the requested content and is finally delivered to the actual content source to trigger the content transmission. A content client initiates the resolution process via a `Consume` message containing the ID of the desired content. The primary resolution procedure follows the same “*provider route forwarding*” rule described in section 4.2.3 (i.e. the `Consume` message will be further forwarded to its provider(s) if the CMF cannot find the content entry in its local CRF repository). In case a tier-1 domain is not aware of the content location, then the request is forwarded to all its neighbouring tier-1 domains until the content consumption request is delivered to the identified content source. If the content is not found after the entire resolution process, an `Error` message is returned to the requesting content client indicating a resolution failure.

We define two distinct content resolution stages:

- **Uphill** – the forwarding of a content consumption request from the local CMF “up” along the provider route until it reaches a domain whose CMF has the record entry for the requested content ID.
- **Downhill** – the forwarding of the content consumption request from the domain whose CMF has the record entry of the requested content ID “down” to the explicit content server that hosts the content.



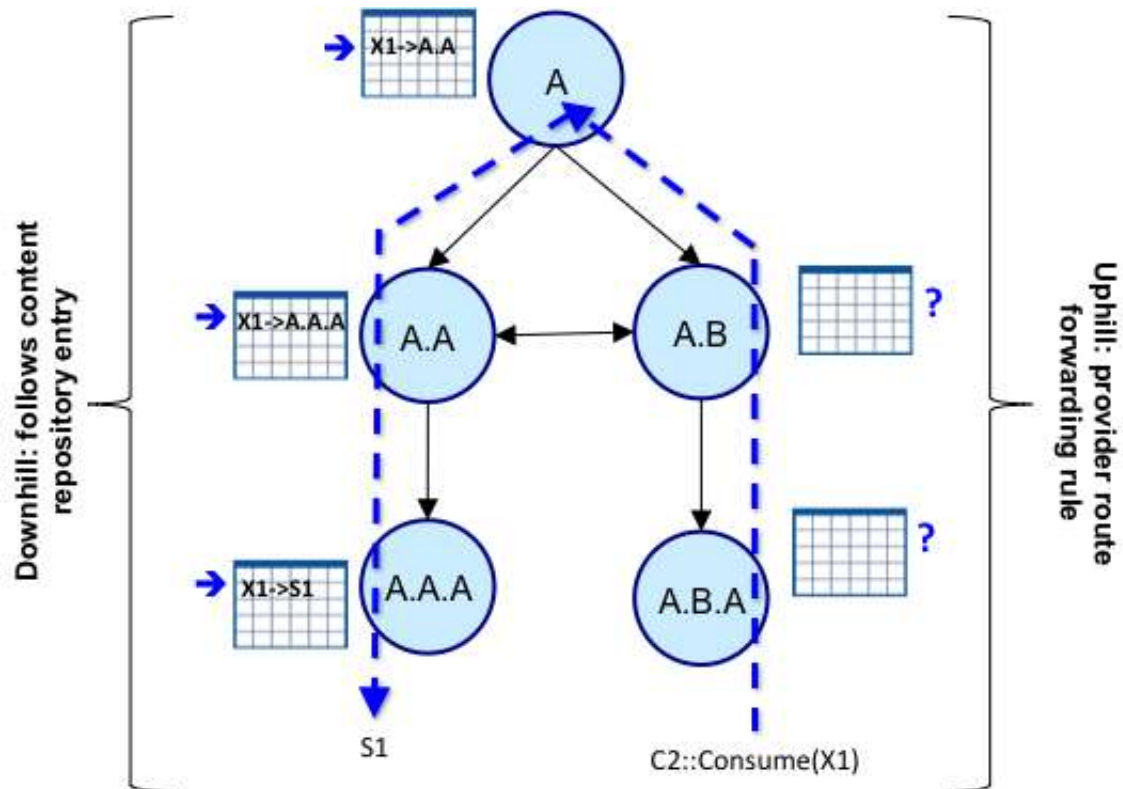


Figure 4-6: Content resolution based on provider route forwarding rule

The two resolution stages are illustrated in Figure 4-6 where content client **C2** at domain **A.B.A** requests for content **X1**. It first sends the `Consume` message to its local CRME. In this example, domain **A.B.A** does not have the record for **X1**. Thus, the `Consume` message is forwarded up to its provider (i.e., uphill) following the provider route forwarding rule (cf. section 4.2.3). This process repeats until the CRME in domain **A** receives the `Consume` message. Domain **A** has the record for the requested content (i.e., content found) and thus, at this point, we enter the downhill stage of the resolution process. The `Consume` message will now be forwarded based on the Next-Hop information in the content management repository rather than following the provider route forwarding route. In the example, the downhill path for the `Consume` message is **A** to **A.A** to **A.A.A** to **S1**.

We further illustrate the process in UML form in Figure 4-7. Note that once the requested content is found, the CRME simply gets the next domain hop instead of forwarding the request to its provider.

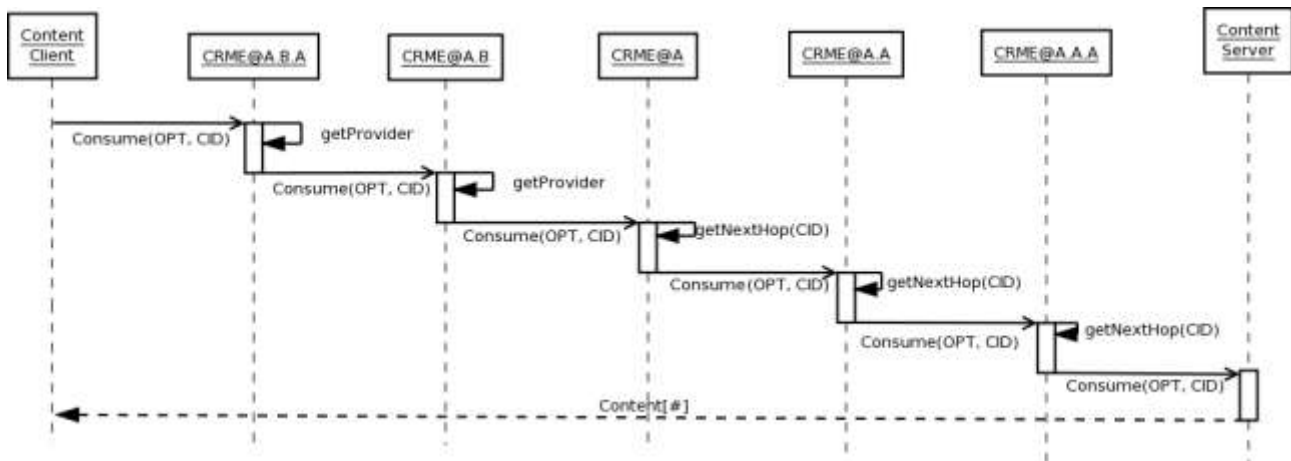


Figure 4-7: UML representation of the content resolution process

Similar to the publication process, *scoping functions* can also be applied in the resolution process, either embedded in the request from a content client or actively issued by a CRME for route optimization purposes during the content delivery phase. Such a function allows a content consumer to indicate preferred ISP network(s) as the source domain of the requested content. Specifically, a content client may use the `INCLUDE` option in `Consume` messages, which carry one or multiple IP prefixes (or domain names) to indicate where he/she would like to receive the content<sup>1</sup> from. Since a set of explicit IP prefixes for candidate content source is carried in the `Consume` message, the corresponding resolution process becomes straightforward: each intermediate CRME only needs to forward the request (splitting required in the presence of multiple non-adjacent IP prefixes) towards the targeted IP prefix(es) directly according to the underlying BGP routes. In case multiple inter-domain routes are available towards a specific prefix, the most explicit one will be followed, as is consistent with today's inter-domain routing policy. In Figure 4-8, content client **C1** issued a `Consume` message for content **X1** indicating its preference for content source in domain **A**. This `Consume` message is then explicitly forwarded towards **A** from **B** following the underlying BGP routing, but without splitting it to **C** despite that a copy of **X1** is also accessible from **C**'s customer domain **C.A**. This scoping-based content resolution path is illustrated with the solid line in the figure.

The *filtering* function in content resolution operations has a complementary effect to *scoping*. Instead of specifying the preferred networks, the content consumer has the opportunity to indicate unwanted domains as possible sources of the desired content. The filtering function is enabled via the `EXCLUDE` option in `Consume` messages. It is important to note the fundamental difference in resolving content consumption requests with *scoping* and *filtering* functions. In contrast to the *scoping* scenario in which a content consumption request is explicitly routed towards the desired IP prefix(es) according to the BGP route, in the *filtering* case, each request is routed based on the business relationship between domains (similar to content publication operations). Consider again Figure 4-8 with content client **C2** requesting content **X1** with the exclusion of domain **C**. Note that since it is multi-homed, the request will be sent randomly to either **A.B** or **B.A** (assuming there is no overriding local policy). For illustration, consider the case where domain **B.A** is chosen (see the dashed line in the figure). However, at the tier-1 level, domain **C** is excluded when resolving this request even though a copy of content **X1** can be found in the customer domain of **C**.

<sup>1</sup> Strictly speaking, it is not always required that ordinary content consumers know the actual IP prefix of the domains they prefer but their local CRMEs may be responsible for translating the "region information" (e.g. domain names) into IP prefixes through standard DNS services.

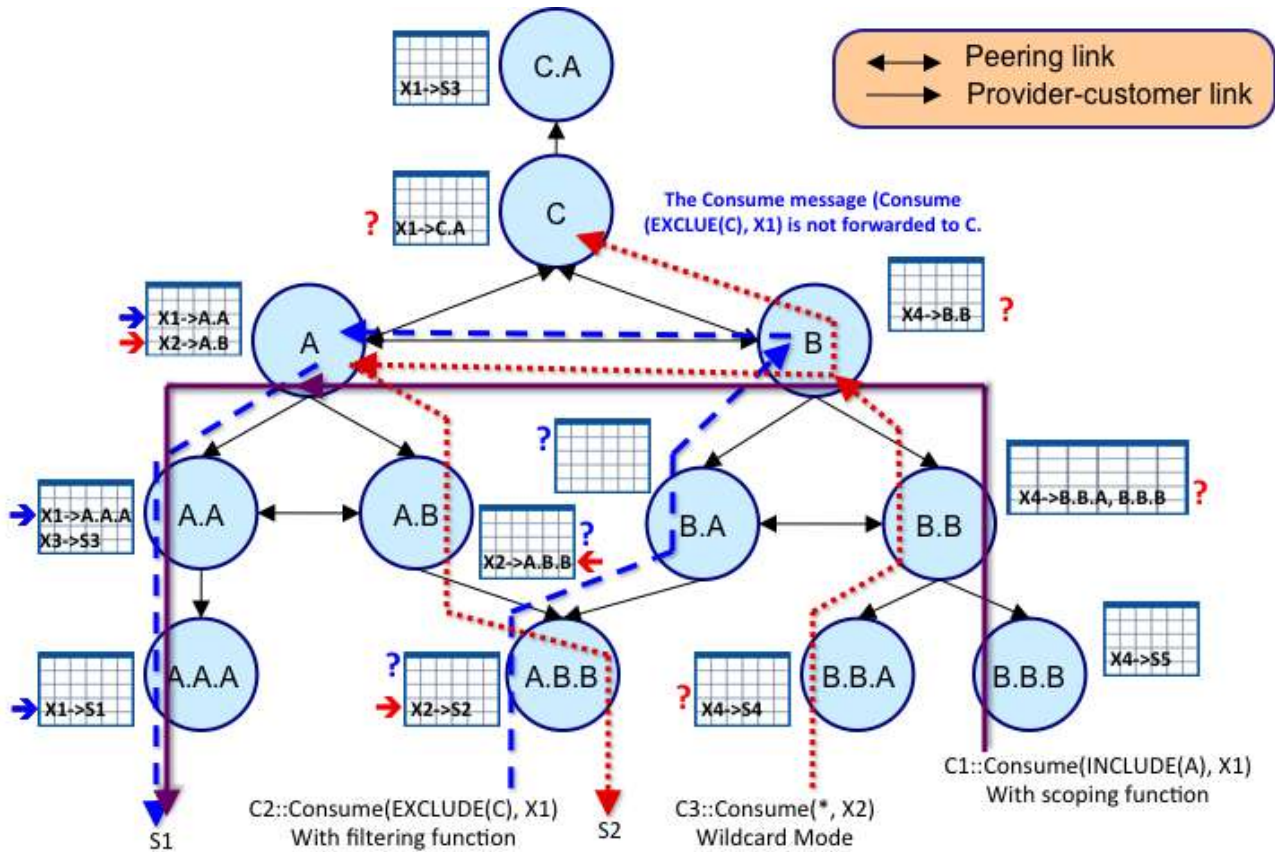


Figure 4-8: Content resolution in different modes

A wildcard in a content consumption request can be regarded as a special case whereby the content client does not have any preference on the geographical location of the content source. The wildcard-based resolution is illustrated in Figure 4-8 via the request from content client **C3** for content **X2** (dotted lines). We see that **B** splits the request to both **A** and **C** at the tier-1 level. Since only domain **A** has the record entry for content **X2**, the request is resolved downhill to **S2**.

Through the illustrations above, we show that bi-directional location-independence can be achieved in the sense that neither content clients nor servers need to know a priori the explicit location (i.e. the IP address of the actual content server and the consumer) of each other for content consumption. In particular, content clients may include *implicit* content scoping / filtering information when requesting content. The content resolution system then automatically identifies the server in the desired “area” that hosts the content. On the content provider side, when a content is published, scoping can be applied such that the content can only be accessed by content clients in the designated area in the Internet. As we will show in the following section, thanks to the multicast-oriented content delivery mechanism, the content server is not aware of the explicit location of the active content clients of that content.

## 4.5 Preparation for Content Delivery

The content delivery process is tightly coupled with the content resolution process and content delivery paths are enforced in a receiver-driven multicast approach. During the content resolution procedure, requests sent by CCs are processed at the mediation plane and are forwarded from one domain to another based on the resolution procedure described above and the CCs *scoping* and *filtering* preferences. While traversing these domains, soft states are installed along the resolution path. Specifically, this is done by configuring the local CAFEs that will be involved in the delivery of the content back from the server to the CC.

When the content is discovered, the delivery path is simply the reverse of the resolution path. The content is delivered following the soft states already installed during the resolution stage. This implies that there is a small requirement on maintaining soft states in the COMET system.

The details of the content delivery procedure for this approach are described in D4.2.

## 4.6 Discussions

### 4.6.1 Scalability

The fundamental domain-level hop-by-hop content resolution strategy presented follows a similar style to that proposed in [7]. However, through the new *scoping* and *filtering* functions, our approach provides the necessary flexibility for both content providers and consumers to publish/request content at/from their desired area(s). The scalability of the system, thus, is dependent on the amount of content and the popularity of the content recorded within each CRME, with the most “vulnerable” CRMEs being those that maintain the highest number of popular content entries. This is in contrast with intuition that the most strained CRMEs will be the tier-1 ones, since content publications and requests may often not reach the tier-1 level based on our approach. Again, we take BBC iPlayer as an example where both the content publication and consumption requests are restricted only to IP prefixes from within the UK. In addition to that, local domain policies may also override the default publication route.

Business incentives also present a natural load distribution mechanism for our system. We foresee ISPs charging higher publication tariffs for popular content published at higher tier domains (with tier-1 domains being the most expensive) which are able to be potentially accessed by a higher number of consumers in the Internet. This mechanism forms a business tussle from the content providers’ point of view when provision of wider access is coupled with higher monetary cost. Instead, a content provider may strategically replicate content to multiple lower-tier regional ISPs (by applying scoping functions there) in which they believe their content will be locally popular.

Finally, our system also allows aggregation in two distinct ways. First, as illustrated in Figure 4-5 for **S4** and **S5**, the record for copies of the same content can be merged during the publication process among CRMEs. Second, a block of sequential content IDs should be allocated to *inter-related* content so that they can be published in one single process. This rule exploits the fact that a specific content provider usually offers content with some relationship with each other (e.g. all episodes of a television series, or all football matches in a World Cup event). This allows for coarser granularity in the publication process whereby the content provider can send only one Publish message to publish all the related content. The local CRME still assigns a unique content ID for each content, but the IDs are sequentially connected. The onwards publication process will only involve the entire block of the IDs rather than the individual content records. This alleviates higher tier ISPs from the need to know each content hosted within and under their domain. Now, instead of matching explicitly the content ID in the Consume message, the CRME simply checks if the content ID is within a specific range of published content. The final location of the specifically requested content is actually handled at the *last-hop* domain where inter-related content entries are locally de-aggregated.

We conducted simulation experiments based on a real domain-level Internet sub-topology rooted at a Tier-1 ISP network (AS7018). This 4-tier network topology is extracted (with aggregation) from the CAIDA dataset [14], with explicit business relationships between neighbouring nodes (i.e., we differentiate provider-customer and peering links). Content sources and consumers were randomly distributed in the domains of this topology. According to our results, the average length of the content resolution paths between individual content consumers and resolved content sources is 4.4 domain-level hops, i.e., the content is on average 4.4 ASes away according to the resolution paths. This is a very good result and also consistent with the general observation that Internet inter-domain sessions are of similar length based on BGP routing and the power-law inter-domain Internet topology.

We are also interested in the actual benefit from such inter-domain routing optimization techniques (refer to D4.2 for the detail of the optimization algorithm) for cost-efficient content delivery across the Internet, especially from the view point of tier-1 ISPs that constitute the Internet core. We used the same domain-level topology as previously described for evaluating the corresponding performance. According to our results, the content traffic (in terms of the number of media sessions) traversing higher tier 1 and 2 ISPs can be reduced by 8.7% through peering route switching, and in particular by a substantial 28.1% in tier-1 ISPs. This is beneficial given that less traffic traverses tier-1 domains through relatively long paths.

#### **4.6.2 Security**

There are different facets in terms of security considerations. First, the communication amongst the CRMEs that form the backbone of the framework must be secured. We assume that each CRME is capable of authenticating their neighbouring counterparts following the exchange of public keys during the establishment of inter-domain relationship (ISP-ISP SLA). Standard third-party security infrastructure may also be used (e.g. using digital certificate) to further ensure the identity of CRMEs.

The current Internet uses IP addresses as both identifier and locator. In all communications, host location is always exposed and thus opens to potential malicious attacks (e.g. Denial-of-Service attack). Location independence is deemed necessary to improve security. As detailed, our architecture achieves bi-directional location independence by pointing to the “implicit” next hop information (e.g. IP prefix) towards the target rather than using explicit addresses associated with content servers.

Furthermore, new proposals on content-centric networks focus on securing the content via self-certified content names with cryptographic features [1][7][41][42]. These measures can be seen as complementary to our architecture and can be accommodated without conflict.

## 5 Server-awareness in COMET system

### 5.1 Introduction

The COMET system includes the server-awareness capability. Server-awareness is becoming increasingly important in today's Internet as content is being replicated by various means (e.g., CDN surrogate servers, P2P dissemination of content etc.). In such content ecosystem, it is often the situation where a specific requested content (especially the popular one) is "hosted" at various locations. As such, equipped with the server-awareness capability, the COMET system provides an additional feature in providing users the requested content from the best available server instead of the current Internet whereby a request is usually resolved to one specific server even if the server is already overloaded.

Due to the different paradigm of the two approaches in COMET, separate server-awareness mechanisms are designed. The following sections describe both of them.

[REDACTED]

[REDACTED]

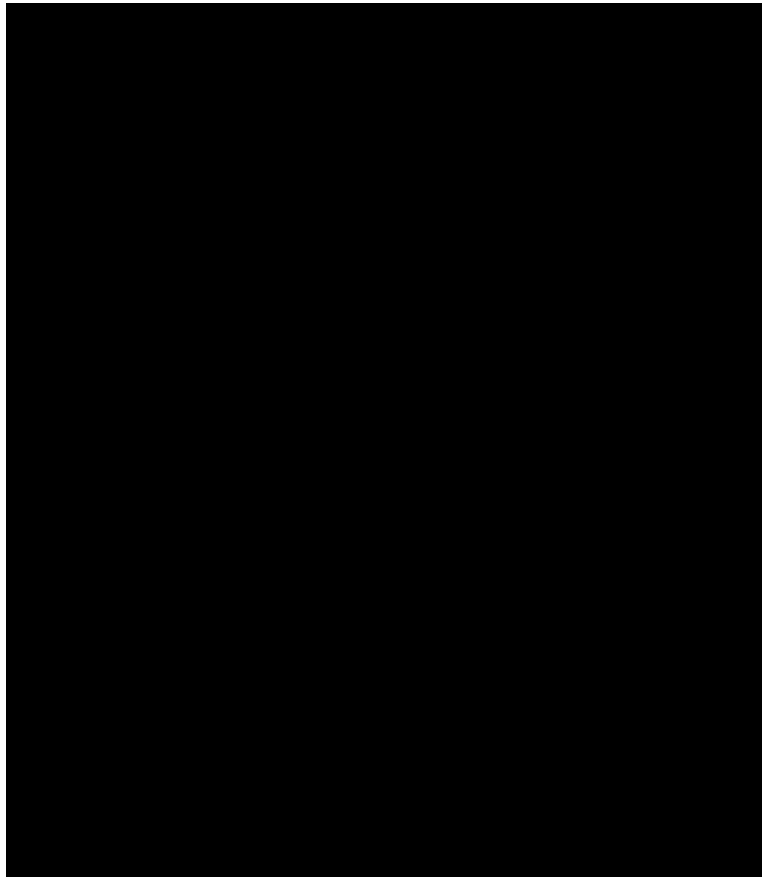
[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[illegible]



[Redacted text line]

[Redacted text line]

[Redacted text line]

[Redacted text block]

[Redacted text line]

[Redacted text line]

[Redacted text line]

[Redacted text line]

[Redacted text line]



[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

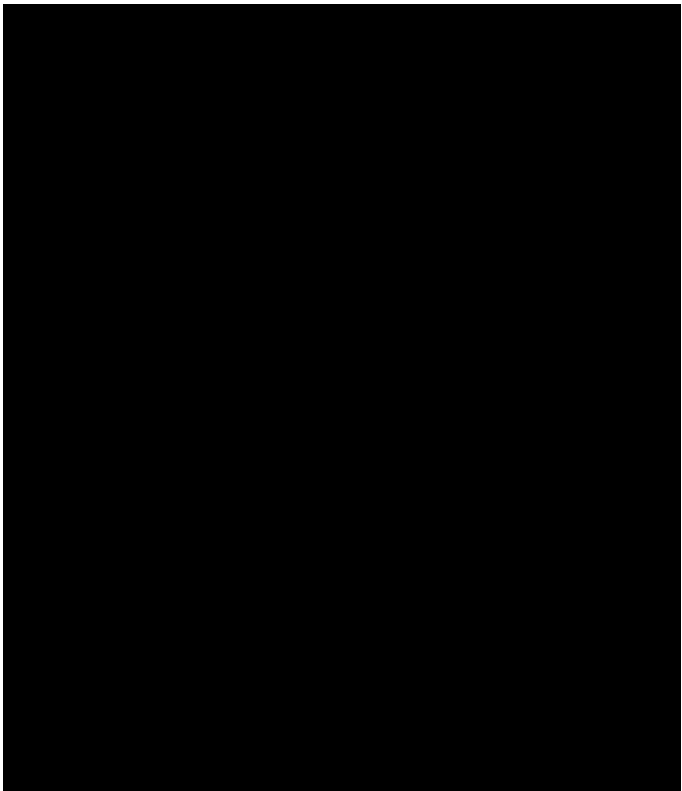
[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]



[Redacted text line]

[Redacted text block]

[Redacted text line]

[Redacted text block]

[Redacted text block]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]		[REDACTED]	
[REDACTED]	[REDACTED]	[REDACTED]	
[REDACTED]	[REDACTED]	[REDACTED]	
[REDACTED]	[REDACTED]	[REDACTED]	

<div></div>	<div></div>
<div></div>	<div></div>

## 5.2 *Server-awareness in the Coupled Approach*

In this section, we describe the server-awareness in the coupled approach. The scheme is compatible with the domain-level hop-by-hop content resolution approach in the coupled approach to optimally identify the content server that hosts the requested content. To enable server-load aware content resolution, we present a technique for disseminating necessary server load information across domains.

The content resolution procedure follows the gossip-like communication model described in chapter 4 which routes content consumption requests from content consumers in a specific manner in order to identify the content source. With the server-awareness support, if multiple copies are available at different content servers, the one with the least load can be optimally selected based on the up-to-date knowledge of server load at individual domains. To enable such a feature, we propose an auxiliary information dissemination technique for (scoped) sharing of dynamic server load information in a multi-domain environment.

### 5.2.1 Design Overview

As described in chapter 4, the proposed content resolution and delivery infrastructure leverages the domain-level hop-by-hop content-based resolution approach whereby a content object is first published by the content provider and discovered through a gossip-like communication model when the content is requested. We now describe the *server-load aware* content resolution process given that when multiple copies of the same content are discovered, the best one is selected based on an optimized server selection process in multi-domain environments. Such a feature has been widely investigated in overlay CDNs (e.g. [17]), but not in native content-centric network platforms at Internet-wide scale.

For this purpose, each CRME maintains an enhanced content record repository for resolution purposes at Internet-wide scale. This repository maintains an entry for each registered content item for public access. Each entry maps a content identifier (ID) to relevant information on all the corresponding sources hosting the content within and “under” the domain of this CRME. Content resolution is achieved through the routing of content consumption request messages between CRMEs residing in each domain, according to their business relationships i.e. provider-customer or peer-peer. In a similar manner, a separate dissemination mechanism across multiple ISP networks on the up-to-date load information of individual content servers is designed for assisting optimized content resolution operations. This is in contrast to the conventional CDN overlay environments where the owner of the CDN infrastructure is able to directly obtain server load information in order to optimally map incoming content consumption requests to specific servers under its control.

### 5.2.2 Dissemination of Server information

We illustrate the content publication process via Figure 5-7 which depicts the *domain-level* network topology with each circle logically representing a domain with a CRME. The process is similar to that described in section 4.3 with small adaptations.

A content provider/owner first registers a new content item to be publicly consumed to its local CRME by issuing a *Register* message. A globally unique content ID will be assigned to the registered content item. We assume that content providers and consumers by default know how to contact their local CRME (e.g. through local configuration). The CRME is then responsible for publishing it outside the local domain to ensure successful discovery by interested consumers in the Internet. The CRME achieves this by transmitting *Publish* messages following the domain-level *provider route forwarding* rule (i.e., each CRME forwards the *Publish* message to its counterpart in the *provider domain* until it reaches a tier-1 domain CRME). Here, a typical *Publish* message carries the content ID and server location where the content item is actually hosted. The information on server location can be represented as “server-ID@domain”<sup>2</sup>, but not in terms of explicit IP address. We illustrate the *Publish* message path in Figure 5-7 (the dashed lines). Each CRME receiving this message updates its content record repository by creating a new entry containing the content ID, the server location information and next-hop domain information (i.e., the neighboring domain from where the *Publish* message has been received). Note that *Publish* messages do not carry server load information which is separately disseminated among CRMEs, and hence the server load information is not applicable upon the arrival of a *Publish* message. Also, the repository is now extended to accommodate server load condition.

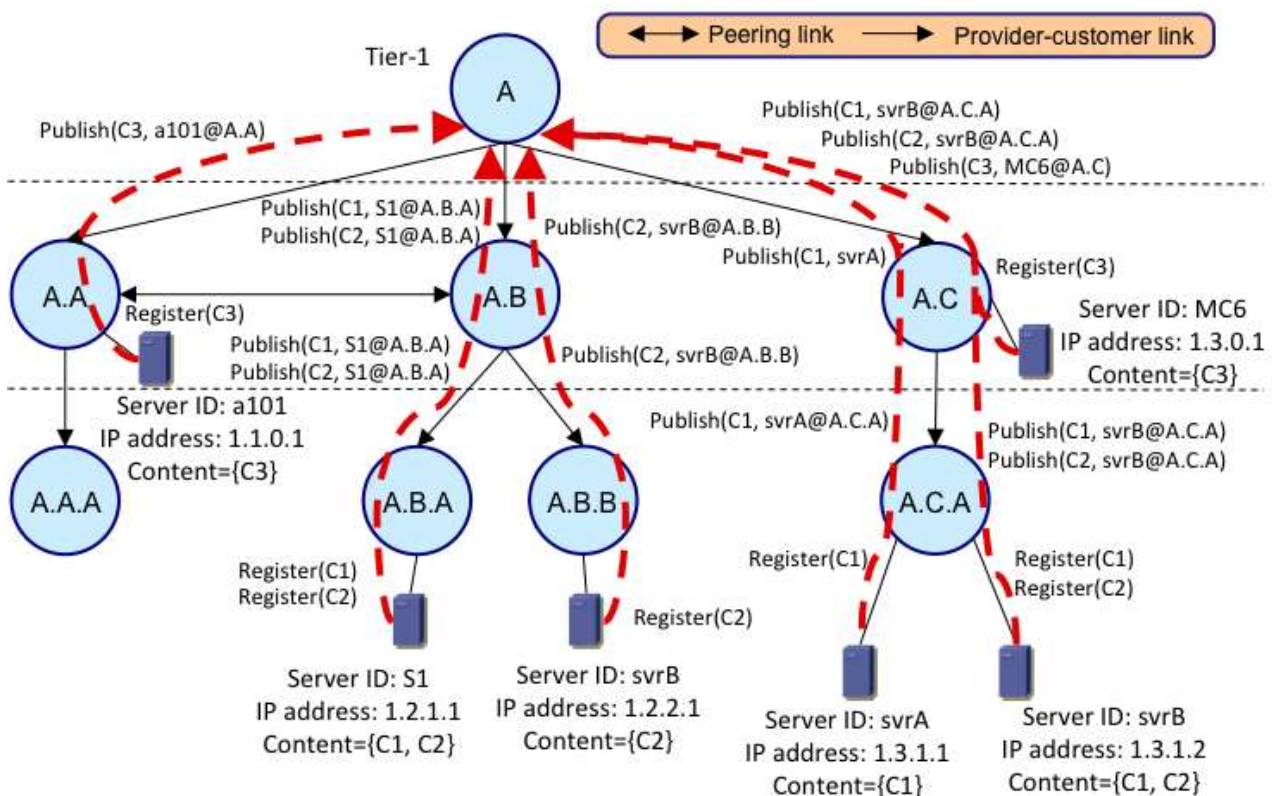


Figure 5-7: Content publication forwarding process

Each CRME will update its content record repository upon the reception of *Register* and *Publish* messages. If a *Publish* message is attempting to register a content item already existing in the repository, the CRME will update the existing content entry by adding the new server

<sup>2</sup> This is different with that described in section 4.3 where only the domain information is recorded.

location information obtained from the newly arrived `Publish` message. Thus, each registered / published content item has one unique record entry in each CRME repository at any time, but possibly with multiple server locations. As an example, Table 1 shows how server location and other information is organized in the content record repository regarding content **C1** at the CRME in domain **A**.

Table 6: Content record repository structure

Content ID	Server location	Next-hop	Server load
C1	S1@A.B.A	A.B	
	svrA@A.C.A	A.C	
	svrB@A.C.A	A.C	

Each domain may have its own server naming space and thus server names can be locally, but not globally, unique. A domain can create dedicated name assignment servers that lease and resolve server names, or content server administrators / owners, as the customer of the ISP-operated content centric platform can register their own server name to the domain.

We use `Update` messages that can be processed by the CRME in individual domains for disseminating server load condition information. The `Update` messages can be either issued *periodically* (e.g. every 10 min per update) or in an *event-driven* fashion, meaning that the message is disseminated only when significant server condition change occurs (e.g. sudden surge in server utilization or server approaching overloaded state). In the former case which is the focus of this paper, we may assume limited discrete level of server load condition (e.g. *Low (L)*, *Medium (M)*, *High (H)*) uniformly used across all domains. The frequency of `Update` message propagation from each server can vary between servers and does not have to be synchronized (i.e., no clock synchronization is required).

The propagation of the `Update` message follows the same rules as the content publication operation (i.e., the *provider route forwarding* rule till a tier-1 domain). The message includes two pieces of information – (1) the server name together with implicit location information and (2) the current load condition. Following the previous example, we illustrate in Figure 5-8 the path the `Update` messages follow (i.e., the dotted lines) to update their server load conditions. As we can see, the server load information is only disseminated up to the root tier-1 ISP network, but not necessarily across the entire Internet.



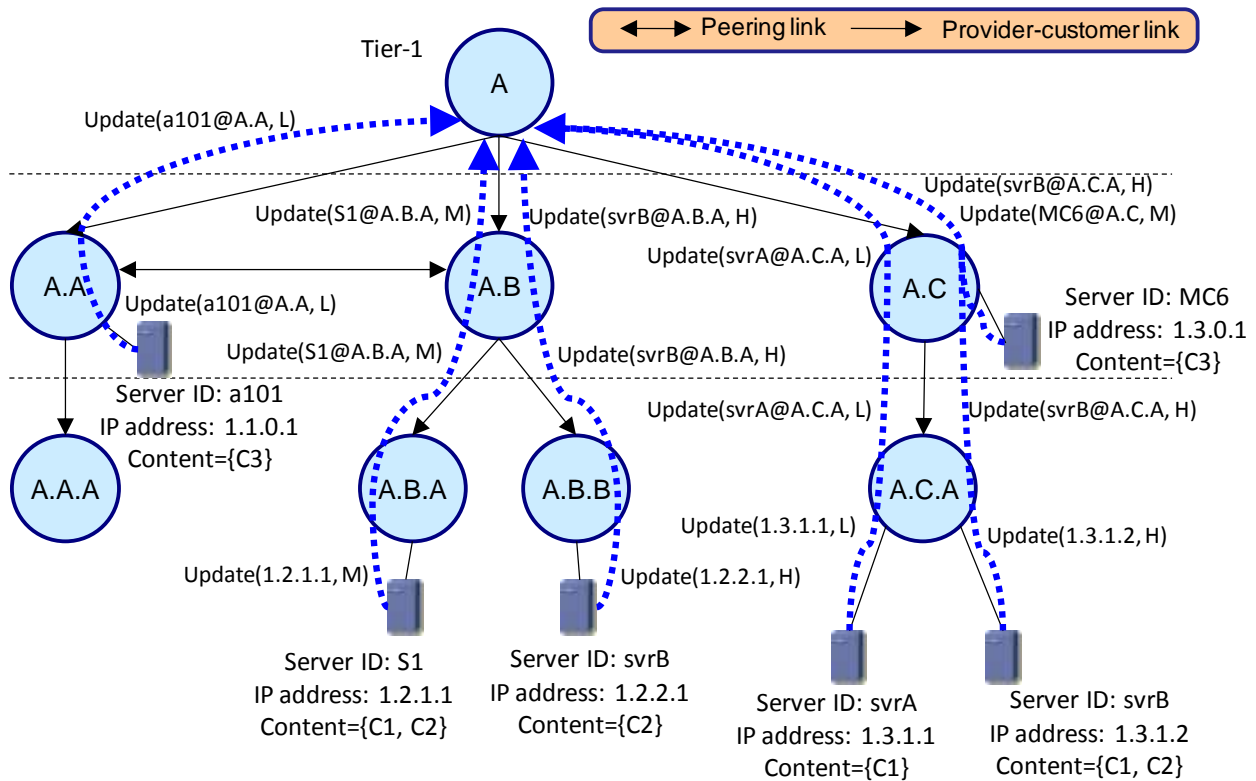


Figure 5-8: Server-load Update forwarding process

The update of the content record repository is on per content ID, but according to the received server load information associated with the server location as the index. We now show as an example the content record repository of the CRME in domain **A** and **A.C.A** after the illustrated publication and a round of updates in Table 7.

Table 7: Content record repository in tier-1 domain A (top) and stub domain A.C.A (bottom)

Content ID	Server location	Next-hop	Server load
C1	S1@A.B.A	A.B	M
	svrA@A.C.A	A.C	L
	svrB@A.C.A	A.C	H
C2	S1@A.B.A	A.B	M
	svrB@A.B.B	A.B	H
	svrB@A.C.A	A.C	H
C3	a101@A.A	A.A	L
	MC6@A.C	A.C	M

Content ID	Server location	Next-hop	Server load
C1	svrA@A.C.A	1.3.1.1	L
	svrB@A.C.A	1.3.1.2	H
C2	svrB@A.C.A	1.3.1.2	H

### 5.2.3 Server-load Aware Resolution

In general, most server selection policies fall into one of the following two: (1) selection aims at achieving equal load distribution amongst servers and (2) selection strives to achieve equal average access delay at the group of servers. It has been shown in [18] that in terms of average delay, both

approaches perform in increasingly similar fashion when the load increases. We use server load as the metric to determine the “best” content server here.

Our server-load aware resolution mechanism on top of the content-centric delivery platform aims to identify content servers for each requesting user in an optimized manner, by taking into account server load conditions. The user initiates the resolution process by sending a content consumption request (through a `Consume` message) containing the content ID to its local CRME.

Following the *provider route forwarding* rule for both `Publish` and `Update` messages, each CRME always has a bird’s eye view of the content hosted and the server load condition within its *own* domain as well as its *customer* domains. When a `Consume` message is received, the CRME checks its content record repository for a matching content record. If none found, the `Consume` message is forwarded to its provider CRME. If multiple matching records are found with different (downhill) locations, then the server with the lowest load condition that hosts the requested content is chosen to serve this request. The `Consume` message is then forwarded onwards to the next CRME based on the next-hop domain information recorded. If there is more than one candidate server having the same low load condition, the CRME may apply additional criteria (e.g. by domain-level hop count information inferred via BGP route). The next-hop CRME will follow the same procedure and forward the `Consume` message based on the next domain hop information in its content record repository until the `Consume` message reaches the server in which case the content flow can be delivered back to the requesting consumer.

We continue to illustrate the resolution process based on our existing example in Figure 5-9 which shows the resolution path from the user to the selected content source. The routing of the `Consume` message on the downhill path follows the next-hop domain information.

- Request for content **C1** from a consumer in domain **A.A.A** (dotted line in Figure 5-9): The first matching content record is found in domain **A**. According to **A**’s content record repository, **svrA** in domain **A.C.A** has indicated low server load while the other two candidate servers both have higher load conditions. Thus, the `Consume` message is passed downhill towards **svrA** via **A.C** (indicated as the next-hop domain in the repository (cf. **Table 7**)). This continues from domain **A.C** to **A.C.A** and finally to **svrA**.
- Request for content **C2** from a consumer in domain **A.A.A** (dashed line): The same operation is executed and in this case, **S1** in domain **A.B.A** is the “best” candidate. Note that there will not be any conflict even though two different servers shared the same name (i.e., **svrB** in **A.B.B** and **A.C.A**) in different domains.
- Request for content **C3** from a consumer in domain **A.C.A** (solid line): For this request, the first matching entry is found in domain **A.C** and only **MC6** (medium load) hosts this content. In this case, despite the availability of a remote server with low load (**a101**) in domain **A.A**, **MC6** is still chosen as the content source to satisfy this request. From this example we can see that it is not our objective to achieve *global optimality* at Internet scale for content server selection, as this requires that all the decision making processes be made at tier-1 domains. Effectively, this is also a trade-off between optimality of server load conditions and the content delivery paths between the server and the consumer.

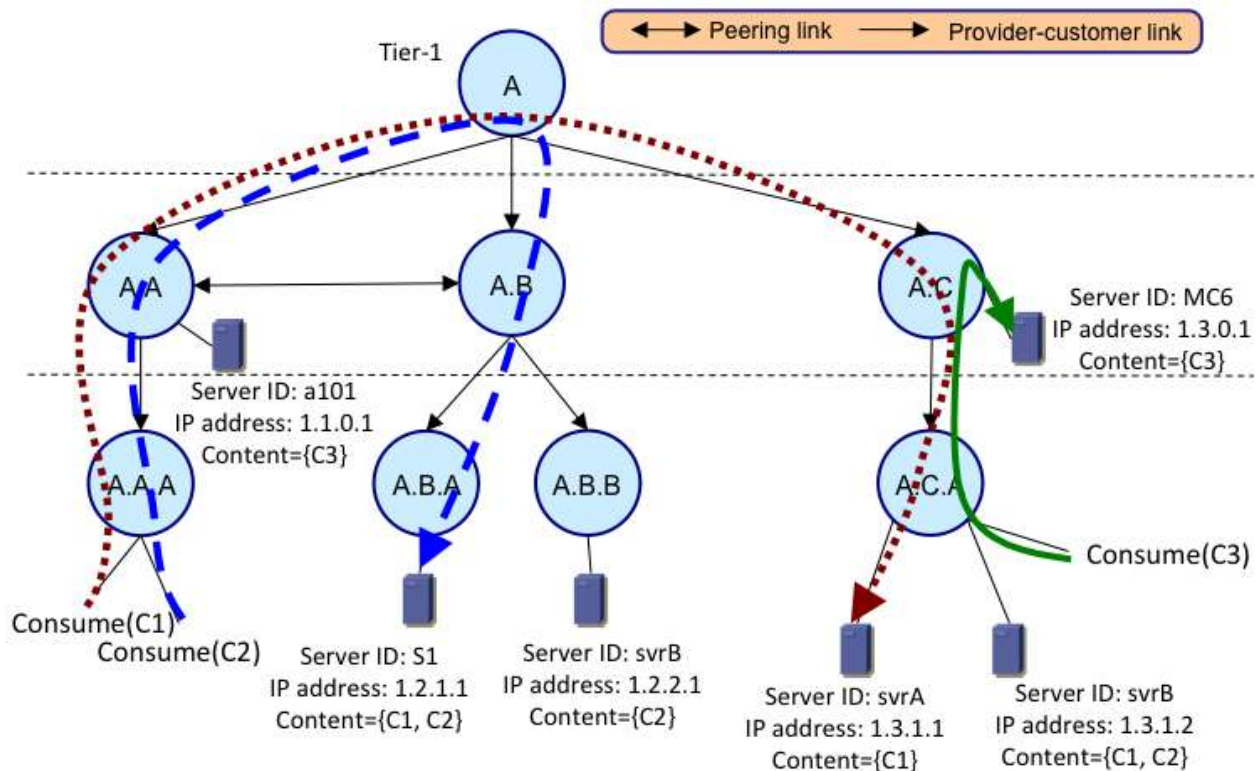


Figure 5-9: Server-load aware content resolution process

If a requested content is not found at the tier-1 CRME, then it will propagate the specific *Consume* message to all its tier-1 peers. An *Error* message is returned to the user indicating a resolution failure if none of the tier-1 CRMEs has the requested content entry in their content record repository.

## 5.2.4 Discussion

As we mentioned previously, there are distinct features and requirements between conventional application overlay based content infrastructures (e.g. standard CDNs) and ISP-operated ones. In CDN environment, the CDN provider is able to have complete control over the overlay infrastructure and of the content servers. In this case, based on the knowledge on the content location and direct monitoring of server load conditions, incoming consumption requests can be strategically directed onto specific content sources. On the other hand, because the overlay infrastructure is fundamentally network-agnostic, the resolution/delivery processes cannot take into account network-level routing awareness.

In ISP-operated content platforms, the key technical challenge is that there is no central entity to perform global content diffusion, but instead individual participating ISP networks need to collaborate with each other in order to fulfil the tasks. One potential technique to ease the interoperability difficulty is through network virtualization where a dedicated content-centric network substratum or plane can be provisioned across participating ISP networks, without radical changes of the current network architecture. As such, dedicated content diffusion functionality can be built within the substratum that spans multiple domains, for instance the realization of CAFEs that are able to natively process content flows based on the dedicated content naming/addressing space. The dissemination technique of server load conditions across domains can be also realized through a dedicated protocol within the content-centric substratum.

It is also worth mentioning the benefits of routing-awareness in ISP-operated content platforms. Thanks to the availability of BGP routing information, optimized path switching becomes possible once the content source has been resolved. There are several solution options to achieve this. On one hand, we may consider to push BGP routing information to the CRME entity so it will be able to perform path switching after the resolution function, but this also requires signalling between

the CRME and local CAFEs for content flow awareness. Alternatively, the functionality of path switching can be implemented at the CAFE side (as CAFEs run BGP in any case) without the involvement of CRMEs. As such, CAFEs should be able to process explicit Consume messages in order to explore the optimized path, instead of signalling between CRMEs. It is not our intention to indicate preferred options, but we only raise the issues for potential further research on the basis of this proposal.

## 6 Evaluation of decision strategies for CAN

In this chapter we present simulation studies which aim at evaluating the performance gain of decision strategies considered for Decision Maker. Those strategies differ in objective functions and assumed reference levels.

### 6.1 Simulation model

The presented simulations focus on content delivery in a large scale model of the Internet (details about the model are included in the Annex A). In this network model, we simulate the arrival process of content requests and content delivery and check whether the content is correctly delivered.

The simulation process runs as follows: the users carry out content requests (Poisson arrival process). Each request is attached to given content. We then compute the server and path selection based on the multi-criteria decision algorithm described in chapter 3.4.6.

The selected server is loaded by one more concurrent connection during a time equal to the content duration. 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 more feasible for modelling unicast UDP-based streaming delivery. At this moment, the load of all the links of the path and the load of the server are checked and, whenever any link or server went beyond the capacity threshold, then we consider that all the connections currently carried by the over-loaded link and/or all the connections currently served by the over-loaded server are served without required guaranties and considered as unsuccessful. Regardless of the state of the path and server, the request is served and the content delivered by the selected path, i.e., both servers and links are considered with enough resources to serve the connections; except for the blocking strategy as discussed below. When a connection finishes, all the links of the connection path as well as the connection server are offloaded by the appropriate values.

The simulations allow us to evaluate the ratio of successful connections. We believe that the use of more information (e.g., information about the load of the server and path) may increase the ratio.

All the simulation tests counted at least  $10^{11}$  served content requests, which is enough to ensure dynamism in the states of servers and links (empty, loaded, over-loaded), i.e., both servers and links changed state many times during each simulation test. In order to confirm this point, we performed tests for checking time range dependence. Moreover, all the 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.

### 6.2 Considered decision strategies

Table 8 presents the evaluated decision strategies corresponding to the optimization objectives taken in the simulations. These strategies cover: (1) random selection of server and delivery of content using shortest path (strategy used in the current Internet), (2) selection of closest server and delivery of content using shortest path (strategy of some CDNs), (3) selection of the least loaded server and delivery of content using shortest path (strategy of some CDNs), (4) selection of the least loaded server and delivery of content using the best path which may not necessarily be the shortest path (strategy assumed in COMET). For forth strategy, we also consider impact of reference levels, which allows to weight importance of particular decision variables.

No.	Decision strategy	Objective function	Parameter(s)	Parameter 1 reservation ( $r_1$ ) and aspiration ( $a_1$ ) levels	Parameter 2 reservation ( $r_2$ ) and aspiration ( $a_2$ ) levels
1	Random server with shortest path (current Internet)	NONE (one server is selected and the shortest	-	-	-

		path obtained from RAE is considered)			
2	Closest server with shortest path (most of CDNs)	$\max_{i \in \{servers, path\}} [\frac{q_1 - r_1}{a_1 - r_1}]$	<i>path_length</i>	$r_1 = \infty$ (e.g. 100) $a_1 = 0$	-
3	Best server with shortest path The best server refers to the least loaded server hosting the requested content	$\max_{i \in \{servers, path\}} [\frac{q_1 - r_1}{a_1 - r_1}]$	<i>serv_load</i>	$r_1 = \infty$ (e.g. 10.0) $a_1 = 0.0$	-
4	Best server with best path The best path refers to the least loaded path which may not necessarily be the shortest path. It is worth mentioning that we only consider the load on the inter-domain links along the considered AS path which can be captured by local SNMEs	$\max_{i \in \{servers, path\}} [\min_{j \in \{1,2\}} \frac{q_j - r_j}{a_j - r_j}]$	$j=1, serv\_load$ $j=2, path\_load$	$r_1 = \infty$ (e.g. 10.0) $a_1 = 0.0$	$r_2 = \infty$ (e.g. 10.0) $a_2 = 0.0$
		$\max_{i \in \{servers, path\}} [\min_{j \in \{1,2\}} \frac{q_j - r_j}{a_j - r_j}]$	$j=1, serv\_load$ $j=2, path\_load$	$r_1 = 1.0$ $a_1 = 0.7$	$r_2 = 1.2$ $a_2 = 1.0$
		$\max_{i \in \{servers, path\}} [\min_{j \in \{1,2\}} \frac{q_j - r_j}{a_j - r_j}]$	$j=1, serv\_load$ $j=2, path\_load$	$r_1 = 0.5$ $a_1 = 0.2$	$r_2 = 1.2$ $a_2 = 1.0$

Table 8: Decision strategies and objective functions

### 6.3 Simulation results

The simulation experiments focused on three main topics. First, we compare 4 decision strategies and evaluate the gain we can reach thanks to awareness of network and server awareness. Next, we evaluate the impact appropriate setting of reference point in multi-criteria decision strategy applied in COMET. In addition, we analyze the impact of admission control on effectiveness of content delivery.

#### 6.3.1 Comparison of decision strategies

As it was said above, content delivery may be considered as unsuccessful by one of two causes: over-loaded path or over-loaded server. Depending on the assumed value of the variables of the system, one cause or the other becomes the bottleneck of the system. We are interested in the work point where both the causes appear simultaneously. Because of this, first we performed tests for very high capacity threshold of the servers and capacity threshold in the link equal to 1 Gbps. Figure 6-1 presents the successful ratio (relation between successful and unsuccessful connections) for the four first decision strategies. As we may observe, the request arrival rate, for which the over-load starts is in the range of 500 request/s.

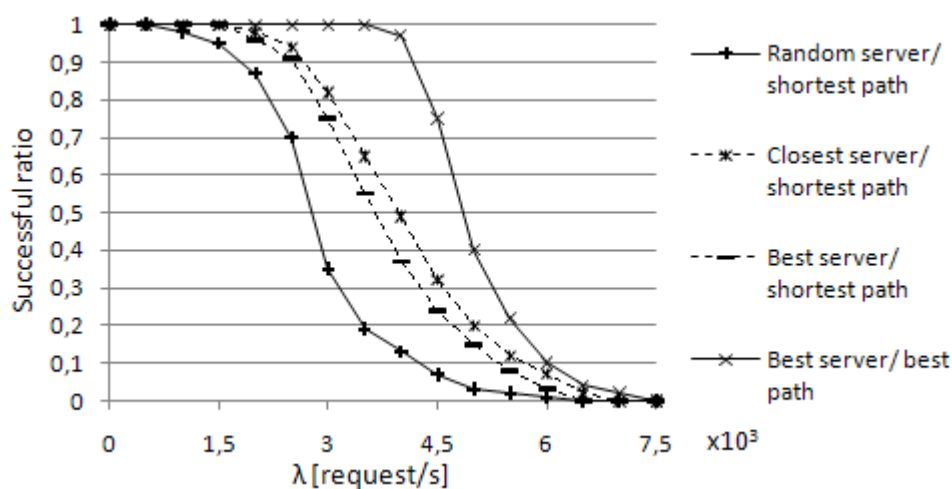


Figure 6-1: Success ratio for server threshold= 108 concurrent connections and link threshold= 1 Gbps. Comparison of 4 decision strategies.

As expected, when the decision process is based on more network information, then the successful ratio increases. In the strategies “closest server/ shortest path” and “best server/ shortest path” one could think that the results should be equal since the server is not the bottleneck in these

simulations; nonetheless, “closest server” strategy ensures that the links are less used and often the selected server is in the same domain of the user and no inter-domain link is loaded by this connection. Because of this, the results for this strategy are a little better.

In the following tests, we searched the value of server capacity threshold for which the servers began to sustain over-load in the range of 500 requests/s. We set the link capacity threshold equal to  $10^8$  Gbps in order to avoid over-load in the links. From one simulation to another, 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 are presented in Figure 6-2. Since the links are not bottleneck in these simulations, the two last strategies offered the same results.

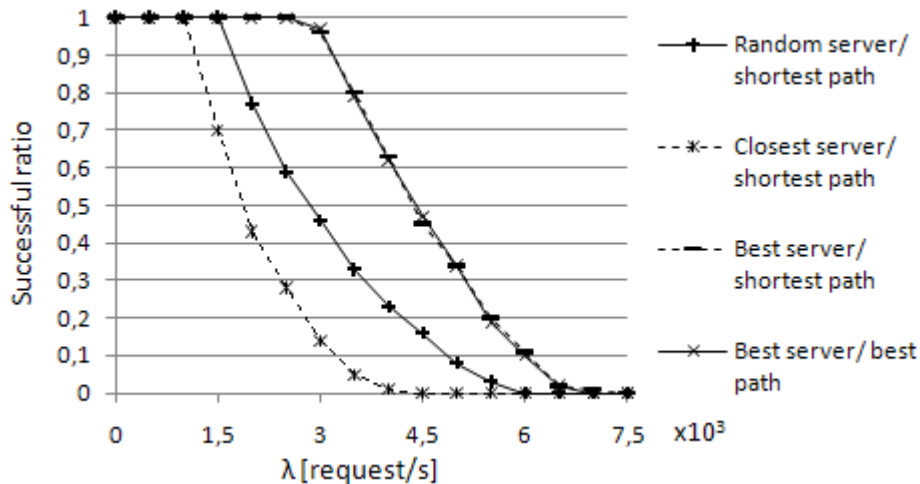


Figure 6-2: Success ratio for server threshold= 200 concurrent connections and link threshold= 108 Gbps.

At last, we performed simulations for server capacity threshold equal to 200 concurrent connections and link capacity threshold equal to 1 Gbps. Figure 6-3 presents the results for this scenario, where both links and servers sustain over-load simultaneously.

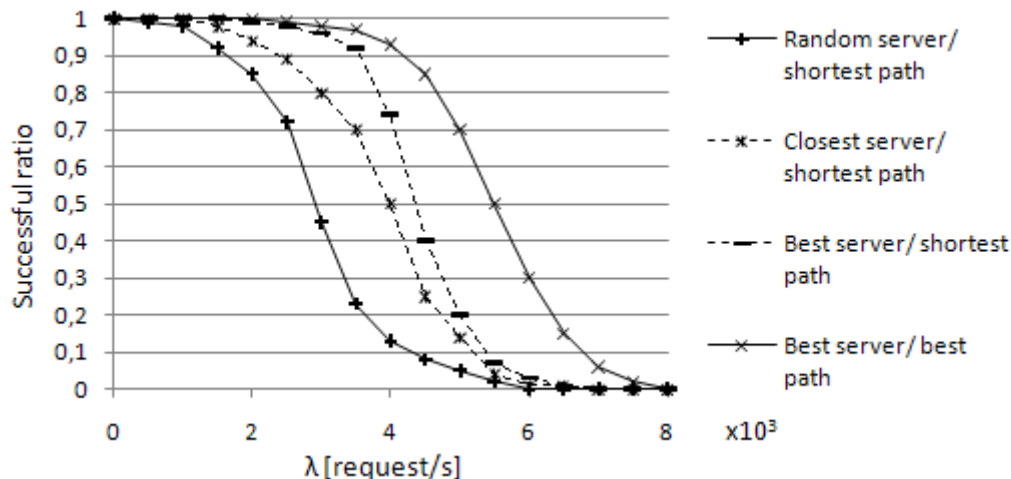


Figure 6-3: Success ratio for server threshold= 200 concurrent connections and link threshold= 1 Gbps. Comparison of 4 decision strategies.

In Figure 6-3 we may observe that the shape of some curves varied from preceding figures. The reason is in the overlap of both over-load effects: in servers and in links. The low confidence interval of the test validates the results.

The results are rather conclusive: the increase of information about links and servers improves the efficiency of the system.



The decision strategy applied in COMET uses the reference points to weight importance of particular decision variable. The values of reservation and aspiration levels (1.2 and 1.0, respectively) are chosen by considering that the bottleneck of the process is the server load and not the path load. Figure 6-4 presents the content consumption success ratio related to different setting of reference levels (see Table 8). The results are also compared with other optimization objectives. For each request the algorithm selects the best pair server/ path on the basis of 100 randomly selected servers and 5 defined shortest paths between the user and each one of the servers (altogether 500 bi-criteria vectors).

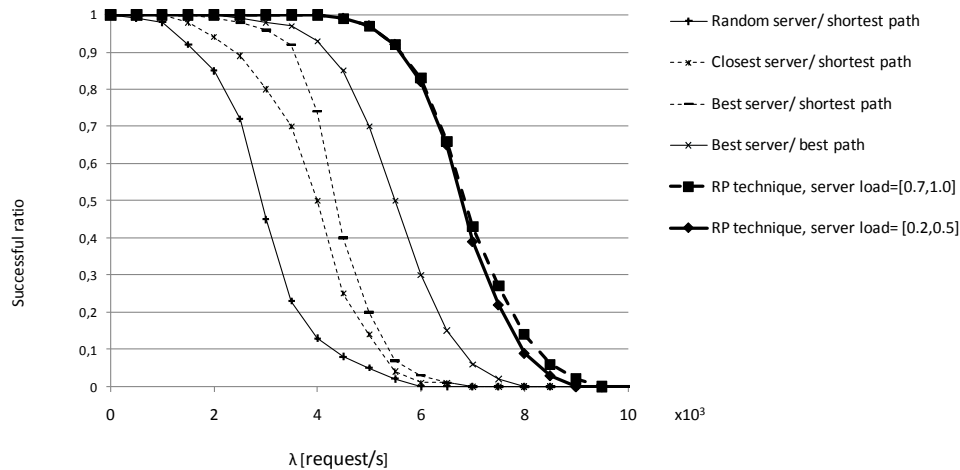


Figure 6-4: Success ratio for server threshold= 200 concurrent connections and link threshold= 1 Gbps. Comparison of 6 decision strategies.

In Figure 6-4 one may observe the gain of the reference point technique when the aspiration and reservation levels are “well-selected”. Let us remark that the values of reservation and aspiration levels in the last two techniques (1.2 and 1.0, respectively) are chosen by considering that the bottleneck of the process is the server load and not the path load.

### 6.3.2 Impact of admission control

In addition to above studies, we analyse the impact of admission control, which prevents the servers and network from overload conditions by blocking the incoming request. Figure 6-5 presents the results when the blocking strategy is applied. This strategy chooses the best server/ best path ( $r_1=r_2=1.0$  and  $a_1=a_2=0.0$ ) but, before accepting the request, the system checks that the new connection will not over-load the server or the path, otherwise the request is rejected and the connection is counted as unsuccessful. The results show the enormous improvement of the efficiency when blocking mechanism is provided.



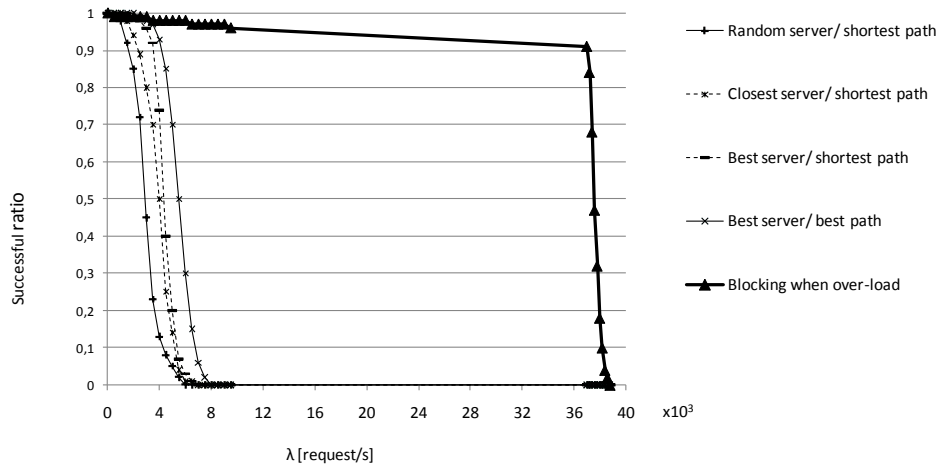


Figure 6-5: Success ratio for server threshold= 200 concurrent connections and link threshold= 1 Gbps. Comparison with blocking strategy

## 6.4 Summary

We demonstrated that the more information the system has, the more effective results it obtains. Moreover, we observed that effective decision algorithms as reference point technique may improve results whenever appropriate reservation and aspiration levels are selected a priori. Special attention requires the blocking strategy since it can decisively increase the effectiveness gain of the system, which is in accordance to the queuing theory.

We are aware that the presented simulation results depend on the assumptions of our model. The range of parameters to be modelled in content Internet is broad, which increases the risk of inaccuracy of the results due to numerous assumptions. Therefore, the presented results cannot be taken as definitive results, but the merit of these simulations lies in the comparison of different strategies based on different information from the network.

## 7 Summary and Conclusions

This document described the final specifications of protocols and algorithms for the Content Mediation System of the COMET architecture. We have shown that the transition to a content-centric Internet requires major changes in the core of the architecture, which through careful steps will guarantee overall stability and efficient content mediation.

Based on the design efforts and the implementation activities, we conclude on the following. For the Decoupled Approach:

- The Path Discovery process, which returns the available paths and their characteristics to Mediation Controller, is self-adapted to the clients' requests as it stores the actually used paths in the local cache and reuses this information for next requests. *Thanks to this we can limit the amount of information kept in Path Storage and, on the other hand, reduce the number of signalling messages related to popular content servers.*
- The complex set of parameters used by Decision Maker motivates us to apply multi-criteria decision algorithm to perform multi-objective optimization. In our approach, we deemed necessary to apply the multi-criteria decision algorithm with the reference points, which allows the operator of COMET system to influence on behavior of decision algorithm. *This owes to the complexity of the system, which in lack of multi-criteria optimisation would not be able to provide accurate information about the optimal delivery path.* In particular, the operator can enforce its own optimization objectives (strategies) defined in a sub-set of decision variables and its preferences by appropriate setting of reference points (upper and lower bounds) for each decision variable.
- The Path Configuration allows flexible enforcing of content delivery paths for each consumption request. Thanks to this, COMET system is able to balance load in the network and assure efficient utilisation of network resources. Although the Path Configuration process is invoked for each consumption request, it is scalable because:
  - statefull operations, e.g. device configuration, packet interception, are limited to the edge CAFEs, while other CAFEs forward packets in a stateless way based on the forwarding keys included in the packet header.
  - for most of content request, the list of forwarding keys is composed based on information stored at the client and server domains. The transit domains are engaged in creation of forwarding list only for the first request served along new path. Normally, it is a rare event happening only when requested path has not been configured yet.
  - The simulation results presented in chapter 6 point out that the path selection strategy applied in COMET (i.e., selection of lightly loaded server over a lightly loaded path) for selection of content server and content delivery path outperforms other strategies used in the current Internet (i.e., random server using the shortest path) and CDNs (i.e., closest server/shortest path and lightly loaded server/shortest path). To evaluate the abovementioned decision strategies, we measure the ratio of successful content requests, where a given content request is successful only when the content server was not overloaded and any link on the path between server and client did not get congested during content delivery.
  - Our extensive simulations show that the COMET system can successfully handle approximately 200% more content requests per second than it would be possible in the current Internet and about 50% more requests per second compared to the most efficient strategy used in CDNs.
  - This effect comes from joint optimization of server and network resources performed by multi-criteria decision algorithm applied in COMET, which redirects content requests to avoid heavy loaded servers or overloaded routing paths. Note that the gain from COMET optimization is especially visible under heavy load

conditions, when some servers or links becomes overloaded. On the other hand even under light load, the COMET optimization avoids server and network congestion by balancing the load between content servers and between content delivery paths.

For the Coupled approach and the related mechanisms, our main conclusions are as follows:

- The scalability of the system, according to the coupled content dissemination approach, is dependent on the amount of content and the popularity of the content recorded within each CRME, with the most “vulnerable” CRMEs being those that maintain the highest number of popular content entries. This is in contrast with intuition that the most strained CRMEs will be the tier-1 ones, since content publications and requests may often not reach the tier-1 level based on our approach.
- In our server-awareness process, following the *provider route forwarding* rule for both `Publish` and `Update` messages, each CRME always has a bird’s eye view of the content hosted and the server load condition within its *own* domain as well as its *customer* domains. This is very important to guarantee efficiency in choosing the optimal server for the content delivery.
- In ISP-operated content platforms, the key technical challenge is that there is no central entity to perform global content diffusion, but instead individual participating ISP networks need to collaborate with each other in order to fulfill the tasks. One potential technique to ease the interoperability difficulty is through network virtualization where a dedicated content-centric network substratum or plane can be provisioned across participating ISP networks, without radical changes in the current network infrastructure.

## Annex A Model for Content Internet

The model of content Internet that we used in the evaluation of decision strategies considers network topology and routing, content servers, content and content requests arrival. The information of the model is stored in three files, which will be the input data of the simulations. The first file presents the topology of the content Internet; to each domain is attached the list of customers, providers and peers as well as the number of prefixes assigned to the domain. Moreover, each domain includes also the list of content servers located in the domain (we consider some special domains that contain a large number of servers as CDN and big content operators). The second file contains a list of all the content identifiers and the servers, where content is stored. The characteristics of each content are the duration and streaming rate. At last, the third file gathers the five shortest paths between each pair of domains (the topology considers inter-domain links and is not aware of intra-domain routing).

The modelled topology of the Internet bases on data provided by Caida [28], which releases each year on January a general topology of the Internet. On January 2011 the Internet topology was obtained on the basis of the routing tables of fifteen nodes around the world observed during five successive days at noon. The source data of the routing tables one may find, among others, in [29] and [30]. By observing the routing tables more times, it is possible to find new links that are visible only when a failure occurs. We divided the domains between Tier-1, 2 and 3 according to the customers, providers and peers connected. The resulted topology counts 36,000 domains (more or less half of them are stub networks) and 103,000 inter-domain links. In addition to the routing information, University of Oregon [29] and RIPE [30] provide also information about the length of advertised prefixes in each domain. The advertised prefixes slightly change on time, so we considered the minimum value during the five days observation period. The histogram of the advertised prefixes is presented in Figure 7-1. Let us remark that, except for some domains, Tier-3 domains have more advertised prefixes than Tier-1 and Tier-2. In our model, we assume that the user population in given domain is proportional to the advertised prefixes.

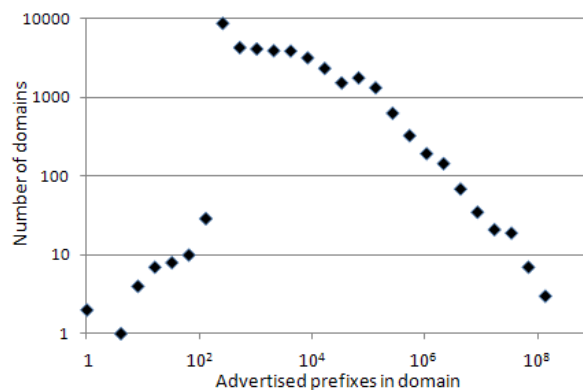


Figure 7-1: Histogram of advertised prefixes in University of Oregon's topology

The last assumption about the topology is the capacity of the links. China Telecom (and many other regions) connects the users by 1 Gbps links [38]. Therefore, we assume 1 Gbps links for connecting Tier 3 to its providers or peers. The inter-domain links between Tier-2 and Tier-1 we assume to be 10 times higher, i.e., 10 Gbps. In our model, we assume that whole capacity of links is dedicated for content traffic.

For the analysis of the distribution of content servers within the domains, we highlighted the 50 largest content video providers and Content Distribution Networks, which are, among others, Level(3), Global Crossing, LimeLight, AT&T, Google and so on [23]. The number of content servers assigned to these domains corresponds to the information provided by the service providers in their webpages. For the rest of domains, we considered that Akamai serves 1,000 domains (as Youtube, QuickTime TV, etc.) with 84,000 servers [37] that mostly are multimedia servers; because of this, we assume that each of the Tier-1 and Tier-2 domains contains 85 servers. We

realize that this assumption is rather weak, but we preferred to assume this value than a random number of servers. The total number of servers in the modeled network is almost 200,000 servers.

The characteristics of the content server, which we are interested in, are the maximum number of concurrent connections that may be served and the maximum number of contents stored. For the first characteristic, both server disk I/O and network bandwidth may be the reason of the limit of concurrent connections [31], [32]. Regardless of the reason, current commercial servers may serve from 50 up to 1,000 concurrent connections [33],[34],[35]. For simplicity purposes all the servers in the network have the same number of maximum concurrent connections but this number varies from one simulation to another as explained in the next paragraph. On the other hand, commercial servers differ much in storage capacity. A medium server may have 600 GBytes content storage capacity [35], which means approximately 100 titles since one High Definition 2-hour film may have a size of 5-8 GBytes [36]. So, by using these data in our model, around 20 million copies of content can be stored.

The features of the video contents we took from filmweb homepage [39] on December, the 1st. We took the 5,000 most popular titles ordered by popularity and with the duration of each film. The mean duration of all the films is around 4100 s. Moreover, to each content we attached a random value of streaming bandwidth comprised between 2,600 and 3,400 kbps. In this range of bandwidth are streamed the videos in Netflix Canadian network [25]. The value of streaming bandwidth is stored in the file with content information.

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 the most approaches of content replication, the number of copies of given content depends on its popularity and follows the Zipf's law which is widely accepted for video distribution in Internet [31],[38],[40]. The skew parameter of the Zipf's formula equals 0.2 as suggested in [38]. In order to have a total number of copies equal to 20 million (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 [24], since a good distribution strategy may definitely increase the efficiency of the system. For simplicity purposes we assumed a random strategy subject to the condition that no more than one copy of given content may be in any server.

Taking into account all the presented assumptions, we are able to model the content Internet. This model is stored into the three aforementioned files that represent the input data of the content delivery simulations.

In the simulations, the request arrival process is unique for the whole network, but each request is attached to specific domain with a probability proportional to the advertised prefixes of the domain stored in the topology file. This way, the domains with more advertised prefixes (larger user population) have higher request intensity. The arrival process for content requests is amply dealt in the literature; it depends on the type of content and type of application, but most of the models suggest for short time scale a Poisson arrival process, for example, [26] for IPTV applications and [27] for Video on Demand applications. Other authors point out modified Poisson process [38] for arrival rate. In our simulations, we took the Poisson arrival model without considering non stationary effects as diurnal/night traffic.

Moreover, each request is attached to specific content by following the Zipf's law for content popularity. We assume one single list of contents, and, therefore, the popularity is unique for the whole network; in fact, users in different domains (e.g., countries) are usually interested in different content.

# Annex B Decoupled Specification Details

This annex includes the specification details that are required for the decoupled approach, but were omitted in the pertinent sections.

## B.1 Content Request Specification Details

This section gathers the details concerning the CC-CME interface as specified in section 3.4.1 (see also section 3.2 and Figure 3-2, for a general description)

### B.1.1 Content Request, Query Datagram Structure

The general structure of the query datagram is shown in the following picture:

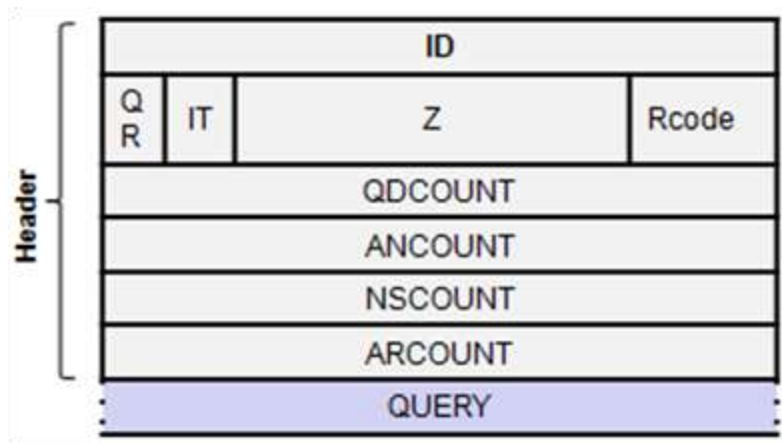


Figure 7-2: Structure of a Content Request (Query)

The header section has the following structure:

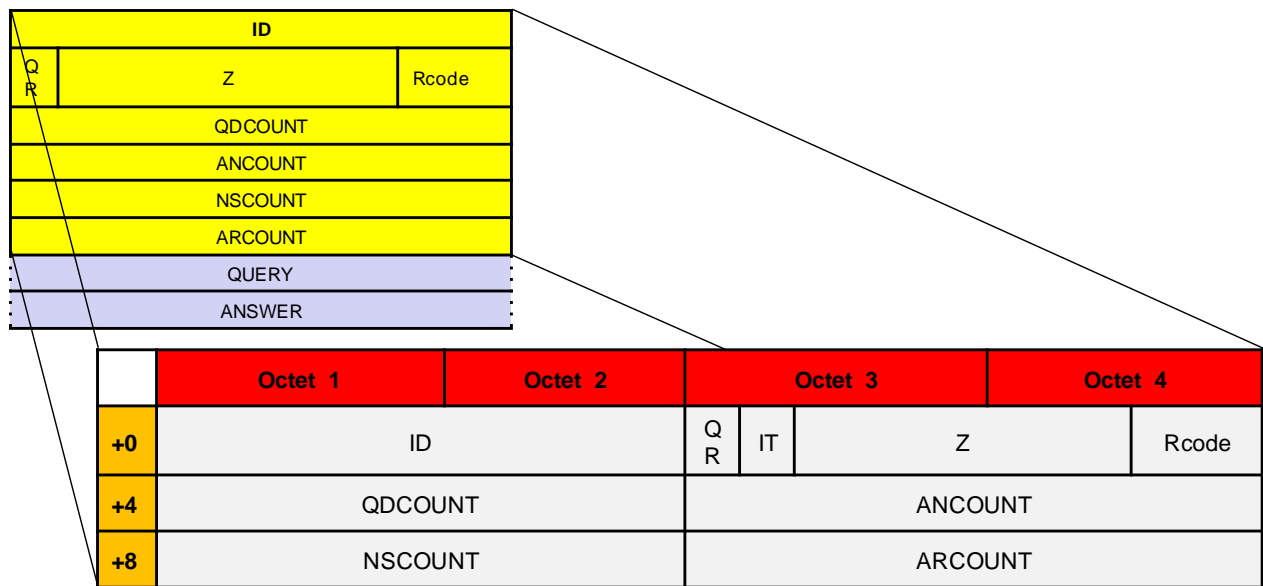


Figure 7-3: Header of a Content Request (Query)

The meanings of the fields are as follows:

Field	Required	Type	Length	Default	Description
ID	Y	Byte	2		Identification for queries and answers
QR	Y	Bit	1		Query (o) or response (1)
IT	Y	Bit	1		Type Content: CID(1) / CN(o)
Z	Y	Bit	10	0	Not used
RCODE	Y	Bit	4	0	Response code from Server. Not used
QDCOUNT	Y	Byte	2	1	Number of queries.
ANCOUNT	Y	Byte	2	0	Number of answers.
NSCOUNT	Y	Byte	2	0	Not used
ARCOUNT	Y	Byte	2	0	Not used

Table 9: Parameters in Header of a Content Request (Query)

**ID:** Identification field used to correlate queries and responses.

**QR:** Identifies the message as a query (o) or response (1).

**IT:** Identifies the type of Content Identifier that is used in the QNAME: Content ID (o) or Content Name (1).

**Z:** Reserved for future use. Set to 0.

**RCODE:** Response Code. Set by the name server to identify the status of the query.

**QDCOUNT:** Question count. Define the number of entries in the question section (set to 1).

**ANCOUNT:** Answer count. Define the number of resource records in the answer section (set to 1)

**NSCOUNT:** Authority count. Number of name server resource records in the authority section. (0 for CC-CME protocol).

**ARCOUNT:** Additional count. Number of resource records in the additional records section. (0 for CC-CME protocol)

The query section has the following structure:

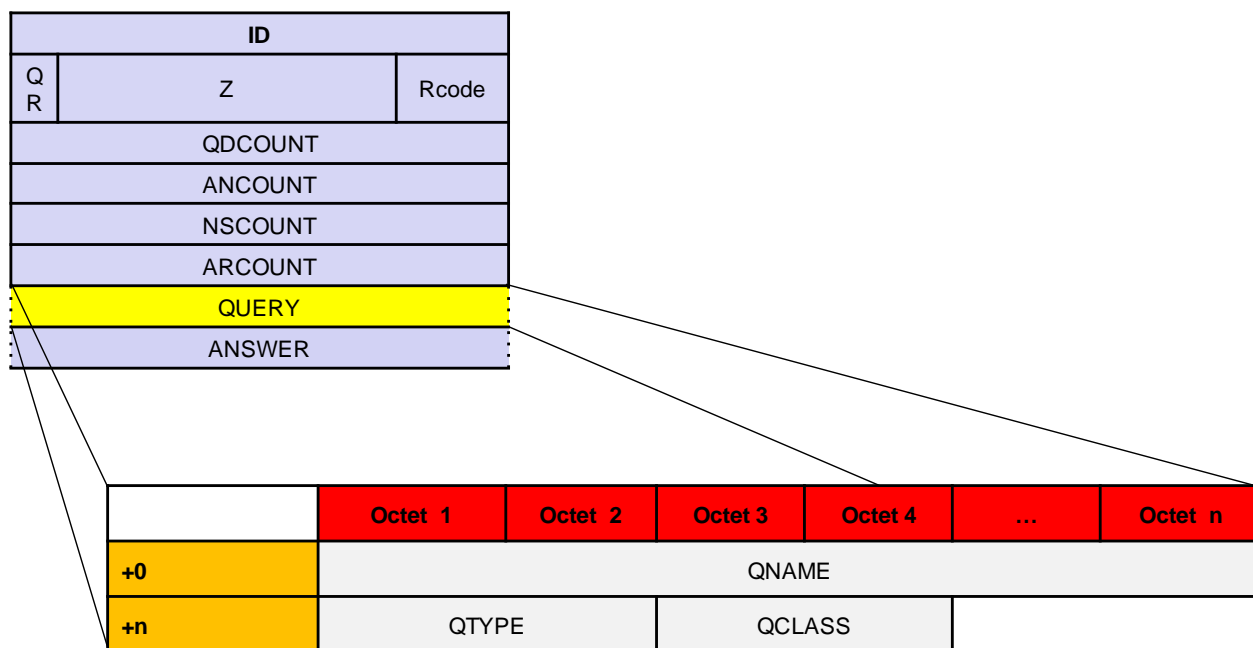


Figure 7-4: Query section of a Content Request (Query)

And the meaning of the fields is the following:

Field	Required	Type	Length	Default	Description
QNAME	Y	Byte	1,N		Size of N(1) + CID or CN (N)
QTYPE	Y	Byte	2	100	Type of entry
QCLASS	Y	Byte	2	1	Class of entry

Table 10: Parameters in Query of a Content Request (Query)

**QNAME:** The first byte of the QNAME indicates the length (hex) of the Content ID or Content Name. The rest of the bytes are the ASCII characters of the Content ID or Content Name of the content the client is asking for.

**QTYPE:** Type of entry. The value for COMET QTYPE could be 100 (to be determined).

**QCLASS:** Identifies the class of an entry. Usually set to 1 (Internet).

### B.1.2 Content Request, Response Datagram Structure

The general structure of the response is shown in the following picture:



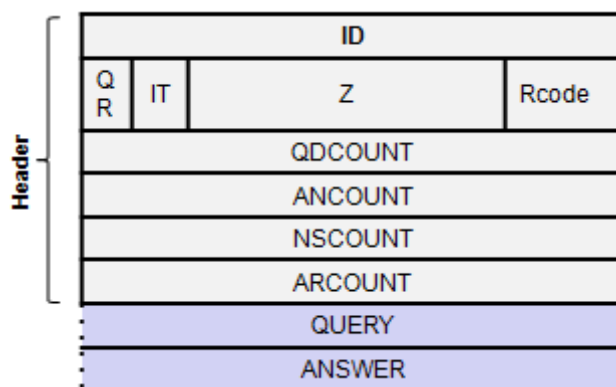


Figure 7-5: Structure of a Content Request (Response)

The structure of the Header section is the same as explained in section **¡Error! No se encuentra el origen de la referencia.** for the Content Request (query).

In the Content Request response, the CME also includes the query section of the content request message.

After the query section, the answer section is included and has the following structure:

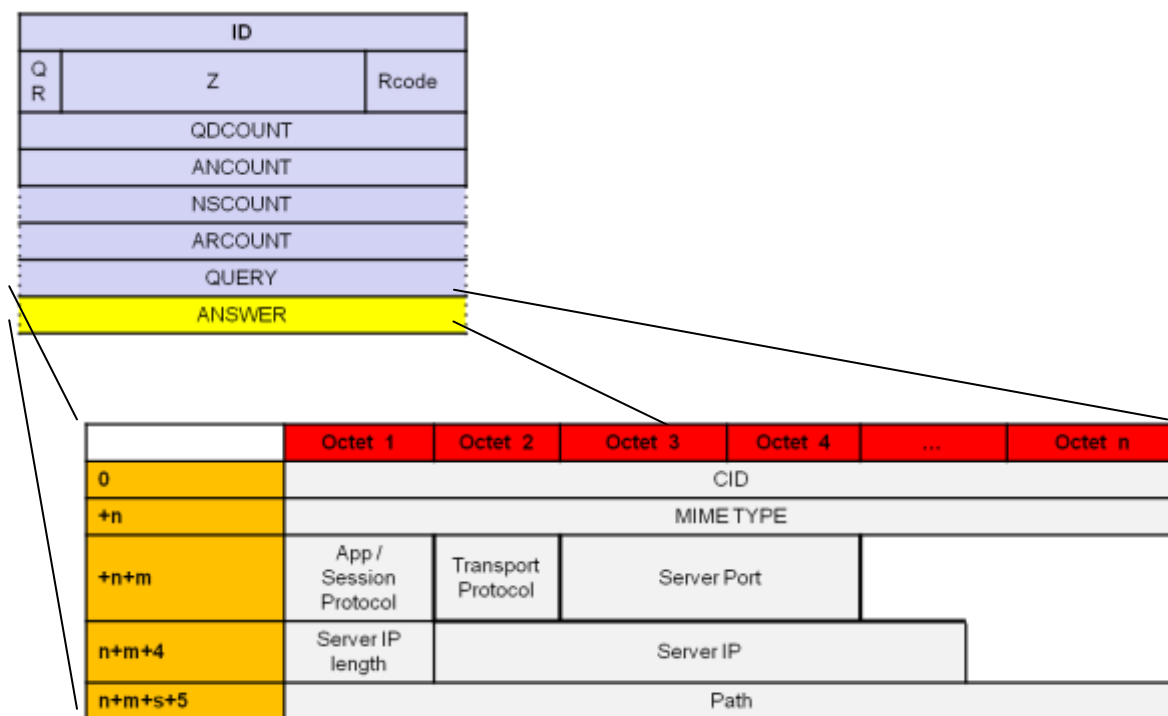


Figure 7-6: Structure of Answer Section in a Content Request (Response)

The meaning of the fields that are included in the response message is the following:

Field	Required	Type	Length	Default	Description
CID	Y	Byte	1,N		Size of N(8) + CID (N)
MTYPE	Y	Byte	1,N		Size of N(1) + Mime Type (N)
ASPROT	Y	Byte	1		App /Session protocol.

TPROT	Y	Byte	1		Transport protocol.
SRVPORT	Y	Byte	2		Server Port.
SRVIPLN	Y	Byte	1		Server IP length.
SRVIP	Y	Byte	2		Server IP.
PATH	Y	Byte	1,N		Size of N(1) + Path (N)

Table 11: Parameters in Answer of a Content Request (Response)

**CID:** The first byte of the CID indicates the length (hex) of the Content ID. The rest of the bytes are the ASCII characters containing the Content ID of the content the client has asked for. This field may contain the value "oxcoXX" (offset to position XX of the CC-CME message) in order to allow the copy of the QNAME from query (in case QNAME is a Content ID). This could bring us a packet size reduction.

**MTYPE:** The first byte of the MIME TYPE indicates the length (hex) of the Mime Type of the content. The rest of the bytes are the ASCII characters of the Mime Type.

**ASPROT:** Application/Session Protocol that must be used by the Content Client in the direct Content Request sent to the Content Server.

**TPROT:** Transport Protocol that must be used by the Content Client in the direct Content Request sent to the Content Server.

**SRVPORT:** It's the port of the server to which the Content Client must send the direct Content Request.

**SRVIPLN:** It's value is 4 (for IPv4) or 16 (for IPv6) depending on the bytes that are required in the Server IP field.

**SRVIP:** IP of the Server in s-bytes format (IPv4 (4) or IPv6 (16) )

**PATH:** The first byte of the Path indicates the length (hex) of the path. The rest of the bytes are the ASCII characters containing the path to follow to find the content inside the server.

The codification of the Transport protocol field (TPROT) is based on the IANA codification of protocol field of the IP header. The following table shows the correspondence:

Hex code	Protocol	Hex code	Protocol
0x00	Hopopt	0x08	EGP
0x01	ICMP	0x09	IGRP
0x02	IGMP	0x11	UDP
0x03	GGP	0x29	IPv6 over IPv4
0x04	IP in IP encapsulation	0x2E	RSVP
0x05	ST	0x2F	GRE
0x06	TCP	0x59	OSPF
0x07	UCL, CBT	...	...

Table 12: Codification of the Transport protocol field (TPROT)

For the codification of the ASPROT field (Application/Session Protocol), the following table will be used:

Hex code	Protocol	Hex code	Protocol	Hex code	Protocol
0x01	http://	0x0C	telnet://	0x17	tcpobex://
0x02	https://	0x0D	imap:	0x18	irdaobex://
0x03	tel:	0x0E	rtsp://	0x19	file://
0x04	mailto:	0x0F	urn:	0x1A	urn:epc:id:
0x05	ftp://	0x10	pop:	0x1B	urn:epc:tag:
0x06	ftps://	0x11	sip:	0x1C	urn:epc:pat:
0x07	sftp://	0x12	sips:	0x1D	urn:epc:raw:
0x08	smb://	0x13	tftp:	0x1E	urn:epc:
0x09	nfs://	0x14	btsp://	0x1F	urn:nfc:
0x0A	dav://	0x15	btl2cap://		
0x0B	news:	0x16	btgoep://		

Table 13: Codification of the Application/Session Protocol Field (ASPROT)

## B.2 Path Configuration Specification Details

This section contains detailed specification of the path configuration process performed in CME. It is invoked when the Decision Process choose the preferred server and content delivery path that should be used for content delivery. The objective of path configuration is the preparation of forwarding plane for content delivery from server's IP address to client's IP address using given transport protocol (TCP/UDP) with given port numbers (at least one port should be provided). Therefore, the path configuration performs two basic actions:

3. it translates the selected path, which has the form of "list of AS numbers", into the list of forwarding rule indicators (keys). The keys are used by particular CAFE to forward packets to next CAFE or directly to client.
4. it configures the CAFE located nearest to the server to intercept IP packets and encapsulate them with COMET header, which includes list of forwarding rule indicators (keys).

This documentation provides specification only for CME level interactions. The following elements are out of the scope of this document:

- specification of CAFE
- interactions between CME and CAFE, although they are mentioned in few places;
- information about resources for provisioning and admission control purposes.

### B.2.1 Information required for path configuration

The path configuration module requires information about CAFEs located inside own domain as well as border CAFEs located in peering domains. These information covers:

- Table 1: – used to find edge CAFE which is responsible for handling traffic to/from a given IP address. Table 1 is needed only in client or server domain.
- Table 2: – used to find CAFEs that transfer or receive packets between peering domains.
- Table 3: – used to find forwarding key which allows to forward packets between neighbouring CAFEs.

Below, we present details of these tables:

**Table 1** allows CME to map the local IP address of client or server into the CAFE handling the traffic passing from/to this address. Formally, we define T1 as:  $(IP\ address) \rightarrow (CAFE\ id)$ . Details of this mapping is following:

- Mapping should operate using IP *prefixes*.
- Result of the mapping must be deterministic (longest prefix matching).
- *CAFE id* can be IP address of the CAFE.

Exemplary T1:

IP prefix	CAFE id
2.2.2.0/24	1.16
194.29.0.0/16	1.17

Note that information stored in T1 depends on configuration of CAFEs inside domain. It should be fixed during domain design and provisioning process. COMET does not change it. In deployment, the COMET system would get this information from other entities, e.g., Domain Management System. In our prototype, we will read this information from configuration file.

**Table 2** allows CME to translate AS path (list of AS numbers) into pair of CAFEs: one belonging to CME's domain and one belonging to peering domain. Formally, we define T2 as:  $T2(\{AS\ path\}, CoS) \rightarrow (source\ CAFE\ id, sink\ CAFE\ id, peering\ CME\ IP\ address)$ . Details of this mapping are following:

- $\{AS\ path\}$  is a sequence of AS numbers.
- CoS is identifier of requested COMET Class of Service.
- Rules for matching against vector  $\{AS\ path\}$ :
  - Find the own AS number (*N*) in the vector  $\{AS\ path\}$  and ignore all values prefixing it, e.g.,  $\{1, 2, 3, N, 7, 8, 9\} \rightarrow \{N, 7, 8, 9\}$ .
  - Match the longest sequence, e.g.,  $\{N, 7, 8, 9\}$  could match against  $\{N, 7\}$ ,  $\{N, 7, 8\}$  or  $\{N, 7, 8, 9\}$ .
- *Source CAFE id* should identify the CAFE sending the traffic on this part of the path. This CAFE is located in own domain and it is used as egress from own domain.
- *Sink CAFE id* should identify the CAFE receiving the traffic on this part of the path. This CAFE is located in peering domain (downstream/sink); it is an ingress to the next domain.
- *Peering CME* is an IP address and port of the CME managing the sink CAFE.

Exemplary T2:

AS peering	CoS	Source CAFE	Sink CAFE	Peering CME
1,2	BE	1.16	2.5	2.2.2.2:9876
2,3	PR	1.16	3.1	3.3.3.3:8765

Note that information in T2 comes from provisioning related with SLA agreements between peering domains. COMET does not change it. In deployment COMET would get this information from other entities, e.g., Domain Management System. In our prototype, we will read this information from configuration file.

- Notice that, T2 include also mapping which provides the information about how to get the current domain from peering domains. The fields are similar to the previous one:  $\{AS\ path\}$  is a sequence of integer values (AS numbers).

- CoS is identifier of requested COMET Class of Service.
- Rules for matching against vector  $\{AS\ path\}$ :
  - For AS path  $\{1, 2, 3, N, 7, 8, 9\}$ , reverse it to  $\{9, 8, 7, N, 3, 2, 1\}$ .
  - Find the current AS number  $N$  in the vector and ignore all values prefixing it, e.g.,  $\{9, 8, 7, N, 3, 2, 1\} \rightarrow \{N, 3, 2, 1\}$ .
  - Match the longest sequence, e.g.,  $\{N, 3, 2, 1\}$  could match against  $\{N, 3\}$ ,  $\{N, 3, 2\}$  or  $\{N, 3, 2, 1\}$ .
- *Source CAFE id* should identify the CAFE sending the traffic on this part of the path. This CAFE is located in peering domain; it is an egress from previous domain.
- *Sink CAFE id* should identify the CAFE receiving the traffic on this part of the path. This CAFE is located in current domain (downstream/sink); it is an ingress to the current domain.
- *Peering CME* is the IP address and port of the CME managing the source CAFE.

Exemplary T2\*:

AS peering	CoS	Source CAFE	Sink CAFE	Peering CME
2,1	BE	2.5	1.16	2.2.2.2:9876
3,2	PR	3.1	1.16	3.3.3.3:8765

**Table 3** allows CME to translate the sequence of CAFE ids into the forwarding rule key used by CAFES to transfer packet. Formally, we define T3 as:  $T3(\{CAFE\ id1, ..., CAFE\ idn\}, CoS) \rightarrow (key)$ . Note that we can distinguish two types of sequences of CAFE ids: 1) peering to next domain, and 2) transit across current domain. Details of the mapping T3 are following:

- $\{CAFE\ id1, ..., CAFE\ idn\}$  is a sequence of CAFE identifiers. The expected length of the sequence is 2, e.g.,  $\{\text{current domain's egress CAFE, next domain's ingress CAFE}\}$  or  $\{\text{current domain's ingress CAFE, current domain's egress CAFE}\}$ . Although, in general case the sequence could be longer.
- CoS is a requested Class of Service.
- *Key* is a sequence of bytes that will be attached to the packets for content delivery over tunnels. Notice that final key attached to the packet is a concatenation of keys valid between peering CAFES. CAFE will use this information to forward packets.
- Note: *Key* value should match the one installed in the CAFE identified by *CAFE id1* in the mapping To, i.e., the key in To must allow for sending traffic along the sequence of CAFES.

Exemplary T3:

CAFE peering	CoS	Key
$\{1.16, 2.5\}$	BE	$\{0xa1, 0xbb\}$
$\{1.16, 3.3\}$	PR	$\{0x0f\}$

Note that information in T3 should be fixed during domain design and CAFE configuration. COMET system does not change it. In deployment, COMET would get this information from other entities, e.g., Domain Management System. In our prototype, we will read this information from configuration file.

When path configuration component configures the path, it stores this information in **Table 4**. This table includes a set of active paths going from edge-to-edge. This table is managed and configured by COMET system. Formally, we define T4 as:  $T4(\{AS\ path\}, CoS) \rightarrow (key, CAFE\ id)$  Details of this mapping are following:

- *{AS path}* is a sequence of AS numbers.
- The mapping should match whole *{AS path}*.
- CoS is a requested Class of Service.
- *Key* is a sequence of bytes.
- *CAFE id* points to the egress CAFE where traffic leaves the server's domain.
- Note: this mapping is used at servers' domains. It could also use statically provisioned paths agreed between ISPs, e.g., a static routing path for AS path {a, b, c} requires an entry in the T4.

Exemplary Table 4

AS path	CoS	key	CAFE id
{1,2,3}	BE	{0xaa, 0x01, 0x17}	1.16
{1,2}	PR	{0xaa, 0x05}	1.16

## B.2.2 Path configuration process

The Path configuration starts after completion of the Decision Process when following information is known:

- AS\_PATH – AS path from server to the client as a list of AS numbers,
- IPC – IP address of the client,
- IPS – IP address of the server,
- FILTER – transport protocol and port numbers (at least one port must be known),
- IPCME – IP address of the server side CME,
- BR – requested bit rate (from traffic descriptor),
- CoS – requested Class of Service.

We distinguish two path configuration procedures. The procedure 1 is used when client and server are located in different domains (this is the most typical situation), while the procedure 2 is used when both client and server are located within the same domain. In order to invoke adequate procedure, the path configurator checks whether IP address of the server (IPS) and IP address of client (IPC) matches prefixes available in T1. If IPS is outside the domain and IPC is inside the domain then the path configurator invokes procedure 1. On the other hand, when both addresses are located inside the same domain, the path configurator uses the procedure 2.

Below, we present the message sequence diagrams for both procedures.

### B.2.2.1 Procedure 1 (client and server located in different domains)

Figure 7-7 presents the path configuration process using procedure 1. It also covers Path Provisioning sub-process which must be performed when requested AS path (list of AS numbers) has not been yet configured and therefore cannot be mapped with T4.

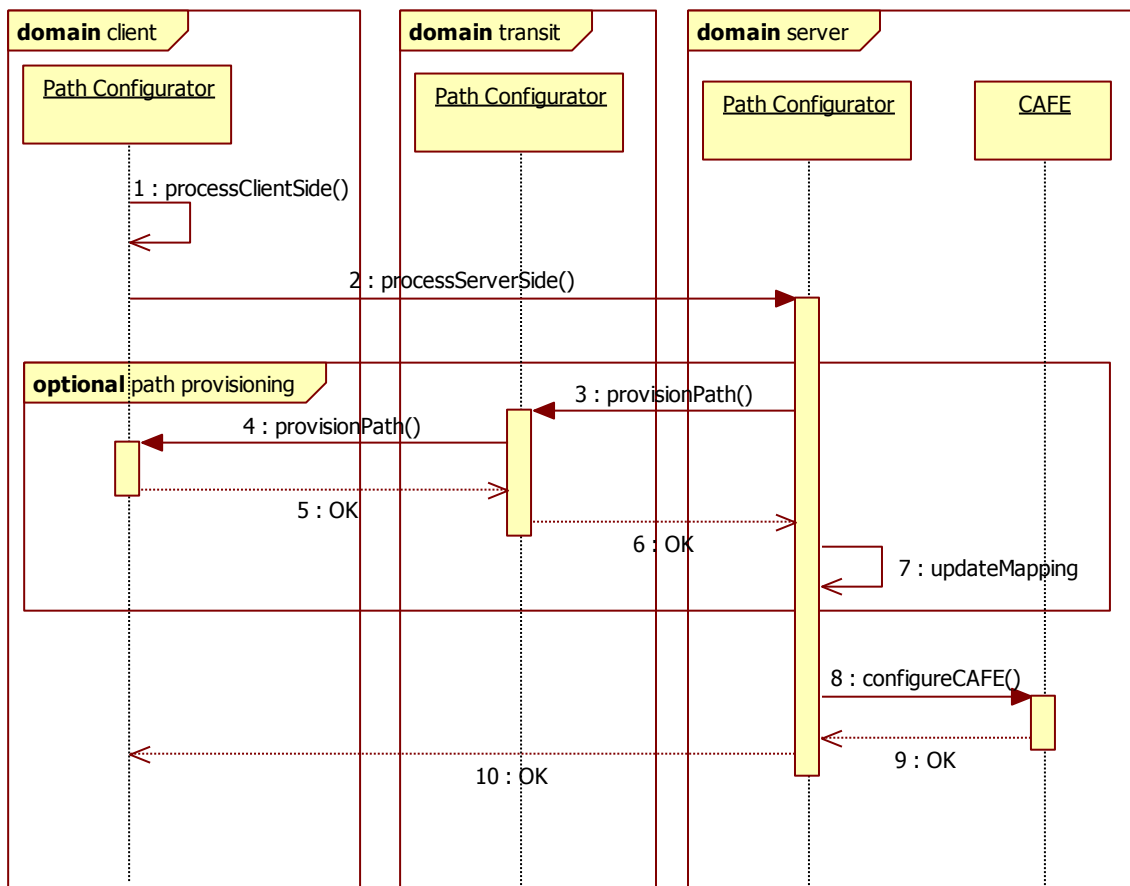


Figure 7-7: Path configuration – procedure 1

Description of the messages:

#### 1. processClientSide()

CME in client's domain starts the Path Configuration process. The objective of client side process is to find key used for packet forwarding from ingress CAFE of client's domain to the client. It performs following actions.

- find CAFE where the client is connected. Set  $CAFE\_DST := T_1(IPC)$
- Use  $T_2^*(AS\_PATH)$  to find ingress CAFE where traffic enters the domain. Set the  $CAFE\_INGRESS$  to the sink CAFE from the  $T_2^*$ . Note: Multiple ingress CAFEs for a given path are not supported in this specification.
- find key which allows to forward packets between ingress CAFE and client CAFE. Set  $KEY\_CLIENT := T_3(\{CAFE\_INGRESS, CAFE\_DST\}, CoS)$ .
- Optionally: set KEY to the session identifier. (This option allows to apply specific forwarding between clients CAFE and client, e.g. multicast, masquerading).
- Merge keys. Set  $KEY := KEY + KEY\_CLIENT$  (concatenation of bytes).
- If client domain requires admission control then perform it for parts: 1)  $CAFE\_DST \rightarrow IPC$ , and 2)  $CAFE\_INGRESS \rightarrow CAFE\_DST$  using requested bit rate (BR) from traffic descriptor (optional).

When any of the mappings cannot be performed, the negative result should be reported.

#### 2. processServerSide(AS\_PATH, IPC, IPS, FILTER, BR, CoS, KEY)

The objective of server side process is to find key used for packet forwarding from server IP address towards the ingress CAFE of client's domain. In this process we distinguish two sub-processes. First, called path provisioning, is invoked only when there is no appropriate path in Table 4. It is responsible for finding key\_path used for packet forwarding from the egress CAFE of server's domain to ingress CAFE of client's domain. The second sub-process is responsible for finding key\_server used for packet forwarding from the server CAFE to the egress CAFE of server's domain.

CME in server's domain continues processing:

- check if required AS path exist in Table 4. Set KEY\_PATH and CAFE\_EGRESS using mapping T4(AS\_PATH, CoS).

In the diagram we assumed that above mapping failed, therefore mediation controller invokes path provisioning sub-process. Such situation occurs when requested path has not been configured yet. Normally, it is a rare event happening only for the first new request along a given path, but it allows us to explain details of Path Provisioning process. Note that paths could also be statically configured following the prearranged agreements between ISPs. In this case, we use CME administration interface to put information to Table 4.

In case when mapping is successful, i.e. the requested path exists, Path Configuration proceeds directly to steps presented after Path Provisioning ends.

### \*\*\* *Optional Path Provisioning starts* \*\*\*

- Select the value of BW\_AGGREGATE for class CoS that will be allocated on the new path. BW\_AGGREGATE is the aggregate value of the bit rate assigned to this new path by Domain Management System. This parameter should be configured in CME.
- Use table T2(AS\_PATH) to find CAFE\_EGRESS (=source CAFE in T2), CAFE\_NEXT (=sink CAFE in T2) and IP address CME\_NEXT of peering CME (owner of sink CAFE).
- Perform provisioning (including long term resource reservation) for path between CAFE\_EGRESS and CAFE\_NEXT for BW\_AGGREGATE in class CoS. This process depends on domain management system. In release 2 we can treat it as optional.
- find key which allows to forward packets between egress CAFE and next CAFE. Set KEY\_NEXT:=T3( {CAFE\_EGRESS, CAFE\_NEXT}, CoS).
- Set KEY\_NEW := KEY\_NEXT.

### 3. provisionPath(AS\_PATH, CAFE\_NEXT, BW\_AGGREGATE, CoS, KEY\_NEW)

Request is sent to the CME\_NEXT where following actions take place (transit domain):

- Set CAFE\_INGRESS := CAFE\_NEXT.
- Check AS\_PATH and select handling for transit domain. Use mapping T2(AS\_PATH) to find CAFE\_EGRESS (=source CAFE in T2), CAFE\_NEXT (=sink CAFE in T2) and IP address CME\_NEXT of peering CME (owner of sink CAFE).
- Perform provisioning for path between CAFE\_INGRESS and CAFE\_EGRESS for BW\_AGGREGATE in class CoS. This process depends on resource management strategy assumed by Domain Management System. In release 2 we can treat it as optional.
- Perform provisioning for path between CAFE\_EGRESS and CAFE\_NEXT for BW\_AGGREGATE in class CoS. This process depends on resource management strategy assumed by Domain Management System. In release 2 we can treat it as optional.
- find key which allows to forward packets between ingress CAFE and CAFE in next domain. Set KEY\_NEXT :=T3( {CAFE\_INGRESS, CAFE\_EGRESS, CAFE\_NEXT}, CoS). This request may be split into 2 parts: ingress CAFE to egress CAFE and egress CAFE to next CAFE.
- Merge keys. Set KEY\_NEW:= KEY\_NEW + KEY\_NEXT (concatenation of bytes).



This step should be repeated until provisioning process reaches client domain.

4. provisionPath(AS\_PATH, CAFE\_NEXT, BW\_AGGREGATE, CoS, KEY\_NEW)

Request is sent to the CME\_NEXT where following actions take place (client domain):

- Set CAFE\_INGRES := CAFE\_NEXT.
- Check AS\_PATH and select handling for client domain (destination).
- Optional: use the KEY value for monitoring the traffic of the AS\_PATH (optional in release 2).

5. OK(AS\_PATH, CoS, KEY\_NEW)

Result is returned to the sender (transit domain).

- Optional: use the KEY value for monitoring the traffic of the AS\_PATH.

This step may be repeated until OK message reaches client domain.

6. OK(AS\_PATH, CoS, KEY\_NEW)

Result is returned to the sender (server domain).

- Optional: use the KEY value for monitoring the traffic of the AS\_PATH.

7. updateMapping(AS\_PATH, CoS, KEY\_NEW, CAFE\_EGRESS)

The value CAFE\_EGRESS must be maintained from step 2.

- Update Table 4 with values {AS\_PATH, CoS, KEY\_NEW, CAFE\_EGRESS}.
- Set KEY\_PATH := KEY\_NEW.

\*\*\* *Optional Path Provisioning finishes* \*\*\*

- Optionally: perform admission control for path AS\_PATH to check whether there is enough bandwidth on the path using BR from traffic descriptor.
- Merge keys from the client side and path. Set KEY := KEY + KEY\_PATH (concatenation of bytes).
- find CAFE where the server is connected. Set CAFE\_SRC := T1(IPS).
- If domain requires admission control then perform it for parts 1) IPS → CAFE\_SRC, and 2) CAFE\_SRC → CAFE\_EGRESS using requested bit rate (BR) from traffic descriptor (optional).
- find key which allows to forward packets between server CAFE and egress CAFE. Set KEY\_SERVER := T3( {CAFE\_SRC, CAFE\_ENGRESS }, CoS ).
- Merge keys to get the final key for forwarding between server CAFE and client. Set KEY := KEY + KEY\_SERVER (concatenation of bytes).

8. configureCAFE(IPS IPC FILTER, BR, KEY, REFRESH\_TIME)

9. OK()

At this moment CAFE should be configured. The value of the REFRESH\_TIME indicates the validity of the soft-state in the CAFE. Each packet sent by server to the client will refresh it. In this specification, we intentionally skip the details about CAFE configuration. The REFRESH\_TIME parameter should be configured in CME.

## 10. OK()

This message confirm client side that path was correctly configured.

### B.2.2.2 Procedure 2 (client and server located within the same domain)

Figure 7-8 explains the path configuration process using procedure 2. This procedure is a simplified version of procedure 1, which is used when client and server are located within the same domain.

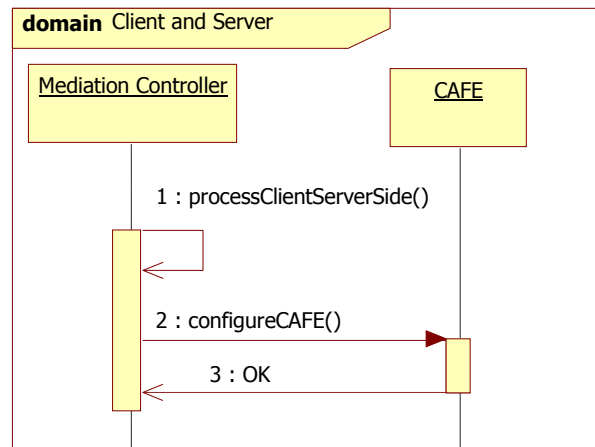


Figure 7-8: Path configuration – procedure 2

Description of the messages:

#### 1. processClientServerSide()

CME starts the Path Configuration process to find key used for packet forwarding from server CAFE to the client. It performs following actions.

- find CAFE where the client is connected. Set  $CAFE\_DST := T1(IPC)$
- find CAFE where the server is connected. Set  $CAFE\_SRC := T1(IPS)$
- find key which allows to forward packets between server CAFE and client CAFE. Set  $KEY\_CLIENT := T3(\{CAFE\_SRC, CAFE\_DST\}, CoS)$ . Note that client and server may be connected to the same CAFE. In this case  $T3$  should contain row with empty key.
- Optionally: set  $KEY$  to the session identifier. (This option allows to apply specific forwarding between clients CAFE and client, e.g. multicast, masquerading).
- Merge keys. Set  $KEY := KEY + KEY\_CLIENT$  (concatenation of bytes).
- Optional: use the  $KEY$  value for monitoring the traffic.
- If domain requires admission control then perform it for parts: 1)  $IPS \rightarrow CAFE\_SRC$ , 2)  $CAFE\_SRC \rightarrow CAFE\_DST$  and 3)  $CAFE\_DST \rightarrow IPC$  using requested bit rate (BR) from traffic descriptor (optional).

When any of the mappings cannot be performed, the negative result should be reported.

#### 2. configureCAFE(IPS IPC FILTER, BR, KEY, REFRESH\_TIME)

#### 3. OK()

At this moment CAFES should be configured. The value of the  $REFRESH\_TIME$  indicates the validity of the soft-state in the CAFE. Each packet sent by server to the client will refresh it. In this specification, we intentionally skip the details about CAFE configuration. The  $REFRESH\_TIME$  parameter should be configured in CME.

Parameters required for Path Configurator are the following: Table 1, Table 2, Table 3, BW\_AGGREGATE, REFRESH\_TIME.

## Annex C Server Awareness Specification Details

This annex includes the specification details that are required for the server awareness mechanisms, but were omitted in the pertinent sections

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

• [REDACTED]

- 
- 

- 

- 

-



## 8 References

- [1] Comet Deliverable D4.2 Final Specifications of the Mechanisms, Protocols and Algorithms for Enhanced Network Platforms
- [2] Comet Deliverable D2.2 High level Architecture of the COMET System
- [3] <http://www.rfc-es.org/rfc/rfc0768-es.txt>
- [4] <http://www.ietf.org/rfc/rfc3650.txt>
- [5] <http://www.ietf.org/rfc/rfc3651.txt>
- [6] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *CoNEXT '09*. New York, NY, USA: ACM, 2009, pp. 1–12.
- [7] T. Koponen, M. Chawla, B-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker and I. Stoica, "A Data-oriented (and Beyond) Network Architecture," in *Proc. ACM SIGCOMM '07*, Kyoto, Japan, Aug. 2007.
- [8] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar and P. Nikander, "LIPSIN: Line Speed Publish/Subscribe Inter-networking", in *Proc. ACM SIGCOMM '09*, Barcelona, Spain, August 2009.
- [9] P. Francis and R. Gummadi, "IPNL: A NAT-extended Internet Architecture," in *Proc. of ACM SIGCOMM '01*, pp. 69-80, San Diego, CA, USA, Aug. 2001.
- [10] I. Stoica, D. Adkins, S. Zhuang, S. Shenker and S. Surana, "Internet Indirection Infrastructure," in *Proc. of ACM SIGCOMM '02*, pp. 73-86, Pittsburgh, PA, USA, Aug. 2002.
- [11] D. Clark, R. Braden, A. Falk and V. Pingali, "FARA: Reorganizing the Addressing Architecture," in *Proc. of ACM SIGCOMM FDNA Workshop*, Aug. 2003.
- [12] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, S. Shenker and I. Stoica, "Routing on Flat Labels," in *Proc. of ACM SIGCOMM '06*, pp. 363-374, Pisa, Italy, Sept. 2006.
- [13] P. Mockapetris, "Domain Names – Concepts and Facilities". Internet Engineering Task Force (IETF) Request for Comments (RFC), RFC 1034, November 1987
- [14] The Cooperative Association for Internet Data Analysis (CAIDA) dataset, online at: <http://www.caida.org/research/topology/#Datasets>
- [15] W. K. Chai et al., "CURLING: Content-Ubiquitous Resolution and Delivery Infrastructure for Next-Generation Services," *IEEE Communications*, pp. 112-120, March 2011.
- [16] B. Ahlgren, et al., "Design Considerations for a Network of Information," *Proc. ReArch '08: Re-Architecting the Internet*, Madrid, Spain, Dec. 2008.
- [17] E. Zegura et al, "Application-layer Anycasting: A Server Selection Architecture and Use in a Replicated Web Service," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, pp. 455-466, Aug. 2000.
- [18] T. Wu and D. Starobinski, "A Comparative Analysis of Server Selection in Content Replication Networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1461-1474, Dec. 2008
- [19] K.S. Pradyumn, "On gradient based local search methods in unconstrained evolutionary multi-objective optimization", in *proc. of the 4th international conference on Evolutionary multi-criterion optimization*, March 05-08, 2007, Matsushima, Japan.
- [20] M. Ehrgott, "Multicriteria Optimization", Springer-Verlag New York, Inc. 2005. ISBN: 3540213988
- [21] 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

- 
- [22] Y. Sawaragi; H. Nakayama and T. Tanino, "Theory of Multiobjective Optimization", Mathematics in Science and Engineering, vol. 176 Academic Press Inc., 1985 ISBN 0126203709.
- [23] 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.
- [24] J. Kangasharju, J. Roberts, and K. W. Ross, "Object Replication Strategies in Content Distribution Networks", Computer Communications, 25(4), Apr. 2002, pp. 367-383.
- [25] 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>.
- [26] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, "Watching television over an IP network". In Proc. of the 8th ACM SIGCOMM Conference on internet Measurement (Vouliagmeni, Greece, October 20 - 22, 2008).
- [27] 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.
- [28] The Cooperative Association for Internet Data Analysis,  
<http://www.caida.org/>.
- [29] University of Oregon Route Views Archive Project David Meyer,  
<http://archive.routeviews.org/>.
- [30] RIPE Network Coordination Centre, "Routing Information Service",  
<http://www.ripe.net/data-tools/stats/ris/ris-peering-policy>.
- [31] H. Bidgoli, The Internet Encyclopedia (Editor-in-Chief), John Wiley & Sons, Inc., Hoboken, NJ, 2004. (3-volumes). ISBN: 0471222046. Volume 2, page 502.
- [32] 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, pp. 141-146, 2009.
- [33] Hitachi, Ltd., "Hitachi VOD Server". ©2010,  
<http://www.hitachi.com/products/it/network/SDP/>.
- [34] NetUP Inc., "IPTV solutions by NetUP: Video on Demand & Virtual Cinema", ©2011,  
<http://www.netup.tv/en-EN/vod-nvod-server.php>.
- [35] VBrick Systems Inc., "VBrick Enterprise Media System". ©2010,  
[http://www.vbrick.com/docs/vbrick\\_datasheet\\_VOD-W-family.pdf](http://www.vbrick.com/docs/vbrick_datasheet_VOD-W-family.pdf).
- [36] 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](http://www.akamai.com/html/about/press/releases/2010/press_031610_1.html).
- [37] Akamai Technologies, Inc., "Facts & Figures",  
[http://www.akamai.com/html/about/facts\\_figures.html](http://www.akamai.com/html/about/facts_figures.html).
- [38] H. Yu, D. Zheng, B. Zhao and W. Zheng, "Understanding User Behavior in Large-Scale Video-on-Demand Systems". In Proc of EuroSys 2006.
- [39] Film Web Inc., homepage: <http://www.filmweb.pl/>.
- [40] K. Gummadi et al., "Measurement modeling and analysis of a peer-to-peer file sharing workload". In roc. Of SOSP. October 2003.



- [41] R. Moskowitz and P. Nikander, "Host Identity Protocol Architecture," RFC 4423, IETF, May 2006.
- [42] M. Walfish, J. Stribling, M. Hrohn, H. Balakrishnan, R. Morris, and S. Shenker, "Middleboxes No Longer Considered Harmful," in Proc. of OSDI '04, pp. 215-230, San Francisco, CA, USA, Dec. 2004.

## 9 Abbreviations

3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorization, Accounting
AS	Autonomous System
ASCII	American Standard Code for Information Interchange
BE	Best Effort
BTBE	Better than Best Effort
BW	Bandwidth
CaCo	CAFE Configurator
CAFE	Content-Aware Forwarding Entity
CC	Content Client
CDN	Content Distribution Networks
CID	Content ID
CME	Content Mediation Entity
CN	Content Name
COMET	Content Mediator architecture for content-aware nETworks
CoS	Class of Service
CP	Content Publisher
CS	Content Server
CRC	Content Registration Client
CRDB	Content Record Database
CRE	Content Resolution Entity
CRS	Content Registration Server
DM	Decision Maker
DNS	Domain Name Server
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
IPTD	IP Packet Transfer Delay
IPLR	IP Packet Loss Ratio
ISP	Internet Service Provider
MC	Mediation Controller
MCDA	Multiple Criteria Decision Analysis
MIME	Multipurpose Internet Mail Extension
MPC	Media Player Classic
MPLS	MultiProtocol Label Switching
NRC	Name Resolution Client

NRS	Name Resolution Server
RAE	Route Awareness Entity
P2P	Peer to Peer
PR	Premium
PS	Path Storage
QNAME	Qualified Name
QoS	Quality of Service
QPI	Qualitative Performance Indicator
RH	Request Handling
RTSP	Real Time Streaming Protocol
SA	Server Awareness
SAS	Server Awareness Service
SIC	Server Information Collector
SLA	Service Level Agreement
SLDB	Servers Load DataBase
SMA	Server Monitoring Agent
SNME	Server and Network Management Element
STREP	Specific Targeted Research Project
TCP	Transport Class Protocol
UDP	User Datagram Protocol
VoD	Video on Demand
VLC	VideoLan Player
WMP	Windows Media Player

## 10 Acknowledgements

This deliverable was made possible due to the large and open help of the WP3 team of the COMET project within this STREP, which includes besides the deliverable authors as indicated in the document control. Many thanks to all of them.