COntent Mediator architecture for content-aware nETworks

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

Deliverable D5.1

Integration of COMET Prototype and Adaptation of Applications

The COMET Consortium

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

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

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

Document Control

Title: Integration of COMET Prototype and Adaptation of Applications

Type: Public

Editor(s): Andrzej Bęben

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

Author(s): Andrzej Bęben, Piotr Wiśniewski (WUT), Lenos Andreou (PTL), Ioannis Psaras, Wei Koong Chai, Stuart Clayman (UCL), George Kamel (UNIS), George Petropoulos (ICOM), David Flórez Rodríguez and Ignacio Conde (TID)

Doc ID: d5.1_v1.3.docx

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
v0.1	30/04/12	Andrzej Bęben	ToC of D5.1
v0.2	15/05/12	Lenos Andreou, George Kamel, Andrzej Bęben, George Petropoulos	First contributions from PTL, UNIS/UCL, WUT, ICOM
v03.	23/05/12	Lenos Andreou, George Kamel, Piotr Wiśniewski	Updated contributions from UNIS/UCL and WUT
v0.4	26/05.12	Andrzej Bęben	First draft of D5.1
v0.5	22/06/12	David Flórez	Contribution about integration of applications
v0.6	22/06/12	Lenos Andreou	Updated contribution about validation tests of decoupled approach
v0.7	22/06/12	George Petropoulos	Updated contribution about integration framework and releases
v0.8	22/06/12	Ioannis Psaras, George Kamel	Reviewed Deliverable, made corrections and added detailed description of validation tests for the coupled approach
v0.9	09/07/12	Andrzej Bęben	Second draft of D5.1
v1.0	11/07/12	David Florez, Piotr Wisniewski, Lenos Andreou	Update about integration of applications and validation tests
v1.1	12/07/12	George Kamel, Ioannis Psaras, George Petropoulos, Andrzej Bęben	Metrics text updated
v1.2	12/07/12	Andrzej Bęben, Piotr Wisniewski	Pre-final version
v1.3	14/07/12	Andrzej Bęben	Final version

Table of Contents

1 Executive Summary

This deliverable focused on the integration of COMET software, the validation of COMET prototypes as well as the integration of applications with the COMET system. Following the COMET architecture, described in D2.2 [1], we focused on two complementary prototypes. The first is related to the COMET decoupled approach prototype, while the second is focused on the COMET coupled approach proof-of-concept. For each of these prototypes, we described the specific integration environment, which covers a set of supporting tools, the integration procedures and the integration testbed built on virtualisation platforms. Moreover, we carried out a number of validation tests corresponding to specific functionalities of particular prototype. Validation of both prototypes followed the bottom-up approach, where we began from validation of particular entities or sub-processes and finished with complex scenarios proving proper inter-working of entities and overall processes. The performed validation tests allowed detecting and fixing of bugs in the developed software, discovering and fixing of inconsistencies in the specifications and identifying bottlenecks in the developed software.

Apart from the above, we also detailed the integration of applications with the COMET system. Based on the analysis of use cases defined in D2.2 [1] and on the demonstration plans from D6.1 [13], we identified four applications that support distribution of live and pre-recorded content in point-to-point, peer-to-peer and point-to-multipoint scenarios.

The content of this deliverable is the following: Chapter 2 includes an introduction. It presents an overview of the COMET architecture, introduces main process and briefly describes the entities designed and implemented for both approaches.

In Chapter 3, we focus on the integration and validation of the COMET decupled prototype. We defined the integration environment, which covers programming languages and libraries used for software development, the tools supporting software integration, i.e. svn, track and hudson, as well as the integration testbed deployed for validation of our prototype. The integration and validation process followed incremental development approach with three releases of the prototype. For each of them, we integrate the developed software and perform a number of validation tests in the integration testbed to verify its functionalities. The identified bugs and open issues were reported to software developers and system designers. In chapter 3, we report tests and results related to the final release of the prototype. The intermediary validation tests are reported in Appendices. The validation process followed the bottom-up approach, which began from the validation of standalone entities or sub-processes related to: (1) content publication, (2) content resolution, (3) server awareness, (4) routing awareness, (5) decision algorithm and path discovery, (6) path configuration, and (7) content delivery. After positive testing of sub-processes, we focused on the overall tests that proved proper content consumption for different types of content in the intra- and inter-domain network scenarios.

Chapter 4 presents the validation approach and results of the proof-of-concept emulator, which is implemented to demonstrate the primary features of the coupled approach. Like the decoupled approach, we validate the correct functioning of the individual entities (CRME, CAFE, Content Client and Content Publisher). The waterfall implementation approach is then presented, followed by the validation of the basic overall operation, namely, *content publication*, *content resolution*, and *content state installation and delivery*.

One of the COMET objectives, as stated in the DoW, is the integration of at least two different applications. Chapter 5 describes how the final COMET prototype can be configured in order to integrate four content distribution services: Streaming, VoD, P2P and Content Streaming Relay (CSR). For each of these services, at least two validation tests cases have been defined, typically one for IPv6 and another for IPv4, and in some cases also for different distribution protocols, like HTTP vs. RTSP. As a result, we proved that COMET is able to manage the most typical means of content distribution in the current Internet. No less important is the fact that the used pieces of software for content distribution (Apache and Tornado Web Servers , VLC, and μTorrent) and consumption (Firefox, VLC and μTorrent) are commercial products that have been used without

any modification, apart from the installation of COMET Content Client into the user equipment. COMET could then be deployed on top of existing content distribution services without adaptations in the final applications.

Following reviewers' recommendations, in chapter 6 we present the set of performance metrics that were identified to evaluate the performance of the COMET system. These metrics correspond to both the overall COMET system performance as well as the benchmarks related to major COMET processes. In particular, the proposed metrics center on: (1) overall content resolution and retrieval process, (2) content publication process, (3) name resolution process, (4) server and network awareness process, (5) routing awareness process, and (6) content forwarding process. For each metric, we proposed the approach to assess obtained results and, whenever it was possible, we identified the reference values based on the analysis of related works on similar systems. The proposed metrics will constitute a base for performance evaluation of COMET prototype, assessment of COMET system scalability and comparison of COMET approaches. In Table 1, we present the summary of identified performance metrics.

Table 1: The summary of identified performance metrics

Target	Metric name	Reference value
Content retrieval	Content Retrieval Latency (CRL)	95 percentile of CRL: < 213 ms, for single domain case < 3 s, as users' tolerable limit
	Content Resolution Time (CRT)	95 percentile of CRT: < 130 ms, for single domain case < 2.5s, as users' tolerable limit
	Content retrieval success ratio (CRSR)	99.9%
	Content resolution signalling overhead, expressed by number of traversed ASes.	No specific reference value is defined because this metric will be used to compare approaches.
Content publication	Number of content records stored on CRE	A single CRE with 1GB HDD would store around 1.26 million content records (assuming the simplest content records)
	Maximum number of users connected to the CP	No specific reference value is defined because this metric will be used to compare CP with HTTP server.
	Maximum publication rate (expressed in [pub/s])	Estimated publication time is around 10-30ms, hence the maximum publication rate would be around 30 - 100 pub/s.
CRE performance	Maximum query rate of root CRE (expressed in [req/s])	The estimated query rate should be around 500 -1 000 req/s per CRE.
	CRE response time (expressed by 95 percentile)	The estimated query time should be around 1-2 ms + RTT per CRE.
SNME performance	Maximum query rate of SNME (expressed in [req/s])	No specific reference value was defined This metric will be evaluated in the testbed to provide data for scalability studies.
	SNME response time (expressed by 95 percentile)	No specific reference value was defined This metric will be evaluated in the testbed to assess the CRT time.
	Number of CS server by a SIC-SNME	A single SIC module is expected to handle dozens of CS.
CME performance	Maximum request rate of CME (expressed in [req/s])	No specific reference value was defined. This metric will be evaluated in the testbed to provide data for scalability studies.
	CME response time (expressed by 95 percentile)	No specific reference value was defined. This metric will be evaluated in the testbed to assess the CRT time.

Content delivery	Lossless throughput of CAFE	No specific reference value is defined, because this metric will be used to compare CAFE with software IP router.
	Number of simultaneous flows	At 1 Gbps port, the maximum number of simultaneous flows should be about 10^5.
	Edge CAFE configuration latency	No specific reference value is defined, because this metric will be used to compare CAFE with software IP router.
	Size of Forwarding Information Base (FIB)	No specific reference value is defined, because this metric will be used to compare FIB in CAFE and IP routers.
	Size of COMET header	The overhead introduced by COMET should not exceed the order of IPv6 header (40B) for the Internet scale network.
	Hop count	After route optimisation, the hop count should not exceed 7–8 hops.
	Bandwidth consumption	No specific reference value is defined, because this metric will be used to compare point-to-multipoint and unicast connections.

Finally, in chapter 7, we summarise this deliverable and outline conclusions and lessons learnt from software integration, validation and adaptation of applications. The conclusions are the following:

1. The performed integration and validation tests related to both prototypes confirm that:
    - Both approaches are feasible for implementation,
    - The functions realized by developed entities and the interfaces between them are conformable to the specifications,
    - The main COMET processes related to content publication, resolution and delivery are properly carried out; allowing for proper content consumption.
2. The incremental development approach with three development cycles including specifications, implementation, integration and validation allowed efficient code development by verifying specification of main components at early stage, providing feedback to the designers about identified open issues, focusing developers, integrators and tester on specific tasks.
3. The proposed integration and validation framework with associated integration tools, i.e. svn, track, hudson, effectively support integration process by keeping track of code changes, simplifying communication between developers, integrators and testers.
4. The validation testbed build based on virtual environment significantly simplifies validation process because it allowed relatively easy repeat tests case conditions. Moreover, we were able to validate COMET software in relatively large network environment consisting of 32 nodes.
5. The bottom-up validation approach allowed identifying bugs and open issues in the developed software and inconsistencies in specification.
6. The lessons from adaptation of 4 applications point out that the COMET system:
    - may be deployed without significant modifications of the consumer terminals or content servers. The required adaptation are: installation of content client module which provide interface to the COMET system, and server monitoring module to provide information about the status of content server,
    - is flexible to interwork with other content distribution services as per-to-peer or point-to-multipoint.

2 Introduction

The COMET project has designed a novel content-aware Internet architecture [1]. It aims to simplify access to any type of content regardless of location and support efficient content delivery in a content-aware fashion. The COMET architecture composes of two planes: the Content Mediation Plane (CMP) and the Content Forwarding Plane (CFP). The CMP is responsible for name and content resolution as well as the preparation of path selected for content delivery, while the CFP is responsible for the content delivery. This is done based on mediation performed by the CMP taking into account the information provided by server, network and routing awareness processes. The COMET architecture was instantiated into two complementary approaches. The decupled approach assumes that the content resolution and content delivery processes are performed in two coordinated phases, while the coupled approach merges the resolution and delivery together into a single hop-by-hop process. Therefore, the mechanisms and algorithms designed for both approaches slightly differ see D3.2 [2] and D4.2 [2] for details. For the same reasons, the entities developed for both approaches are also different, see deliverables D3.3 [4] and D4.3 [5] for details.

The entities designed and implemented for the decoupled approach are the following:

- Content Mediation Entity (CME) is the main entity responsible for mediation. It receives consumer requests and gathers information about the content location, the available routing paths and the server and network status. On that basis, the CME makes decisions on content, server, and path selection. Finally, it configures the edge CAFE located close to the content server to enforce the selected path;
- Content Resolution Entity (CRE) is responsible for storing information about the content and responding to the content name resolution requests during the content resolution process;
- Content Publisher (CP) is used to create, update and delete content records from the CRE database;
- Content Client (CC) allows consumers sending requests to gather content;
- the Server and Network Monitoring Entity (SNME) is responsible for collecting the status of content servers and edge CAFEs. It provides this information to CME;
- Content Server (CS) hosts content and makes them available for consumers;
- the Routing Awareness Entity (RAE) calculates the inter-domain routing paths and provides this information to CME;
- stateless Content Aware Forwarding Entity (CAFE) delivers content from the content server to the consumer through the content delivery path selected by CME.

The entities designed and implemented for the coupled approach are the following:

- Content Resolution and Mediation Entity (CRME) is the main entity responsible for coordination of the publication, resolution and delivery of content;
- Content Publisher (CP) enables content publication. It is also hosting content playing the role of content server;
- Content Client (CC) sends content consumption requests and receives the content.
- stateful Content-aware Forwarding Entity (CAFE) delivers content from the content server to the consumer following content delivery path configured in a hop-by-hop manner during the content resolution process.

In this document, we focus on the integration of developed software, the validation of prototypes related to both approaches as well as the integration of applications with the COMET system. For both prototypes, we defined the specific integration environment, which covers a set of tools supporting integration, the integration procedures, and the integration testbed build on virtualisation platforms. Moreover, we defined and carried out a number of validation tests corresponding to specific functionalities of particular prototype. Validation followed the bottom-up approach, where we began from validation of particular entities or sub-processes and finished with complex scenarios proving proper inter-working of entities and overall processes. The performed

validation tests allowed detecting and fixing bugs in the developed software, discovering and fixing inconsistencies in specification and identifying the bottlenecks in the developed software.

The organization of this document is the following. In chapter 3, we present the integration and validation process of the COMET decoupled prototype. We describe the integration environment, the deployed integration testbed, the development cycles corresponding to prototype releases as well as the performed validation tests, results and conclusions. Chapter 4 focuses on the integration and validation of the coupled prototype. Like the decoupled approach, we present the validation environment, the testing procedures related to the individual entities and overall operations as well as the results and conclusions. In chapter 5, we focused on the integration of applications with the COMET system. We selected four types of applications that are video streaming, video on demand, peer-to-peer (p2p) and content streaming relay. For each application, we present the integration scenario, the validation tests, results, and conclusions. Finally, chapter 6 summarises this deliverable and provides conclusions.

3 Integration and validation of decoupled prototype

3.1 Introduction

In this chapter, we present details of the integration and validation process of the COMET decupled prototype. In section 3.2 we present the integration environment. It covers the description of programming languages and libraries used for software development, the tools supporting integration as well as the deployed integration testbeds. The implementation of the decupled prototype followed incremental development with three cycles covering design, implementation, integration, validation, and software release. The history of system releases is presented in section 3.3. For the software validation, we followed the bottom-up approach, which began from validation of the standalone entities and sub-processes, and finishes on tests related to overall processes. The validation tests, results and conclusions related to the final release of the COMET decupled prototype are reported in sections 3.4 and 3.5. The intermediary validation tests performed for particular releases are reported in Appendixes A, B, and C, respectively. Finally, section 3.6 summarises this chapter.

3.2 Integration environment

3.2.1 Implementation and integration framework

The implementation and integration framework defined for the COMET decoupled prototype includes: (1) all the tools and supporting libraries required for designing, developing, integrating, testing, and deploying the prototype as well as (2) the system tree layout for the COMET software management. These tools were used by the developers to implement their respective components, as well as by integrators, in order to produce a coherent, integrated system. This ensures to some extent that developers and integrators use a common development, building, and testing environment.

The implementation and integration framework covers:

- specification language for describing COMET system,
- programming languages used to develop components,
- the tool chain for compiling and building the source code,
- the version control system for archiving source code,
- an issue tracking platform for debugging, reporting and managing the COMET software,
- continuous integration server for automatically building system's components,
- and finally, the validation testbed used for validating the software.

All tools are described further in the next sections.

Besides the common implementation and integration framework, the COMET consortium defined certain procedures for releasing software components intended for integration into the COMET system, as well as releasing particular versions of the integrated software. These procedures applied to both the development and integration teams of the COMET project.

The definition of the implementation framework and integration procedures was an effort to achieve homogeneity in the delivery of integrated software to all component suppliers involved in the COMET project and assist software component suppliers as well as the software integrators.

3.2.1.1 Specification Language

The COMET architecture is modelled in the Unified Modeling Language (UML) by using the StarUML 5.0 tool [6]. This tool was used to design the interfaces and data structures. The UML diagrams per component are included in deliverables D3.3 [4] and D4.3 [5], where the specification of each component is presented.

3.2.1.2 Programming Languages

Three programming languages were chosen for the development of the decoupled entities. The CME, CP, CRE, SNME and CS are developed in Java 6 (JDK 1.6.0_26) [7]. The C++ language was used for developing the CC and RAE. Finally, the C language was used for developing the CAFE kernel modules. Table 2 presents supporting libraries used for development of particular entity.

Table 2: Libraries per entity

Entity	Libraries	Purpose
CME	netty 3.2.4	NIO client server socket framework
	log4j 1.2.16	Logging framework
	protobuf 2.3.0	Used in interfaces' specification and development
	jsf 1.2 and richfaces 3.3.3	MVC web framework and AJAX-enabled library used in web interface development
	jetty 7.3.0	Embedded web server for CME admin web interface deployment
	hibernate 3, mysql-connector 5.1.15 and c3p0 0.9.12	CME Database (MySQL) connection and resource pooling
	handle 7.0	Handle System library for name resolution
CP	log4j 1.2.16	Logging framework
	jsf 1.2 and richfaces 3.3.3	MVC web framework and AJAX-enabled library used in web interface development
	jetty 7.3.0	Embedded web server for CP's web interface deployment
	handle 7.0	Handle System library for content record creation, update and deletion
CRE	handle 7.0	Handle System library
CC	winsock2	Provide support to send and receive message packets thought sockets.
	ws2tcpip	Provide functions and structures used to retrieve IP addresses.
CS	log4j 1.2.16	Logging framework
	sigar 1.6.4	Cross-platform API for collecting software inventory and monitoring data.
SNME	log4j 1.2.16	Logging framework
	protobuf 2.3.0	Used in interfaces' specification and development
	mysql-connector-java- 5.1.18-bin	SNME Database (MySQL) connection
	jcommon-1.0.17	User interface classes for displaying information about applications and custom layout managers

	jfreechart-1.0.14	Chart Library. Provide support for graphical output types
	jfreechart-1.0.14-swt	SWT implementation accesses the native GUI libraries of the operating system for building graphical interfaces.
RAE	boost:asio 1.49.0	Provides support for network sockets and simplifies implementation of dual stack (IPv4/IPv6)
	protobuf 2.3.0	Used for implementation of RAE interfaces, i.e., inter-RAE and RAE-CME
	sqlite3	Database used for storing routing paths
	log4cxx	Logging framework for C++
CAFE	libnl 1.0	Provides communication between the Linux kernel modules and configuration tools
	protobuf 2.3.0	Used for implementation of CME-CAFE interface

3.2.1.3 Tool Chain

Apache Ant 1.8 [8] is the main tool used to build, deploy and test CME, CP, CRE, SNME and CS entities. It requires a Java Virtual Machine (JVM) to run.

On the other hand, CC requires *minGW* 3.17 [9] and RAE requires *waf1.6.0* [10] for configuring, compiling and installation. Moreover, the compilation and installation of CAFE kernel modules requires Linux kernel headers corresponding to appropriate version of Linux kernel.

The COMET entities also use the following tools (they are included in the repository tree):

- JUnit 4.8.2 for Java unit testing,
- Cobertura 1.9 to calculate code coverage for JUnit tests.

In addition, the some additional tools are required to run COMET system (they are not included in the repository tree):

- MySQL Server 5.1 used in CME and SNME,
- Berkeley database in CRE,
- Sqlite3 database in RAE,
- Python2.7 to prepare RAE configuration files and run CAFE configuration agent,
- VLC 1.1.4 used as a streaming content server, Apache 2 and Tornado web servers used for CS and CSR.

All entities (CME, CP, CRE, SNME, RAE and CS) can successfully run in all operating systems except from the CC, which is Windows-specific, and CAFE, which was developed for Linux.

3.2.1.4 Version Control

The Subversion [11] (svn) version control system is used to archive and version the source code of the decoupled prototype.

The svn repository is organized in 3 categories:

- **Trunk** is the main line of development, originating from the start of the project until now. The system tree layout is presented in Figure 1.

- **Branches** are separate lines of development, deriving from a certain point of the trunk. The main purpose for creating a new branch is for experimenting with or applying major changes in the code, separately from the trunk, which if they work properly, they would be finally merged into the trunk.
- **Tags** are snapshots of the trunk or the branches that the consortium would wish to maintain. In the COMET svn, tags folder contains all releases of decoupled prototype.

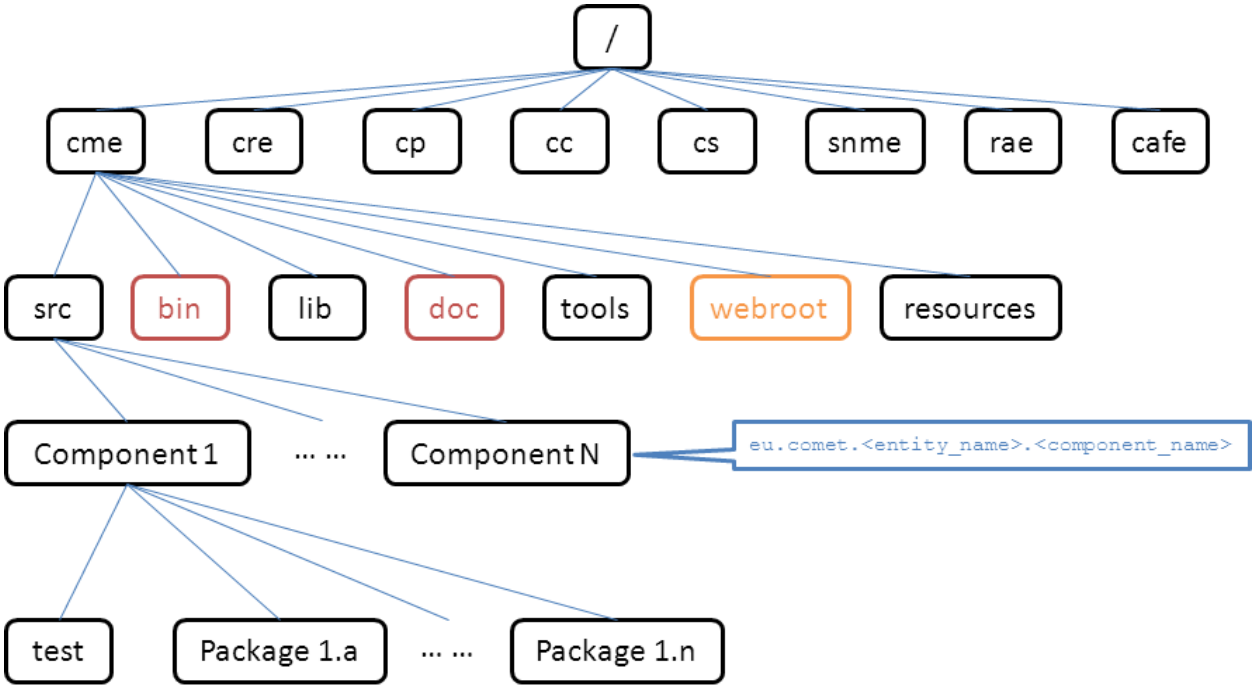

Figure 1: The system tree layout

3.2.1.5 Issue Tracking

TRAC[1] is used as COMET's issue tracking platform, where integrators can organize developers' work, as well as report issues (features and bugs) that need to be resolved. TRAC is also integrated with SVN, providing the ability to track source code changes.

One of the main advantages of TRAC is ticketing. If an issue occurs during the implementation (e.g. a bug) or a new feature needs to be created and added to our software, the reporter (who might be either the integrator or any developer being affected by the issue) has to assign a ticket to the responsible component owner, who will then be responsible for resolving it.

Each ticket has a certain lifecycle, described below and presented in Figure 3:

- **New:** A new ticket is created when an issue occurs. When creating a new ticket, the reporter has to assign it to the responsible developer as well as provide a small description of what must be done or fixed. Moreover, the reporter provide ticket priority, the deadline and target version of the software which will include the new feature or fix of the bug, and other essential attributes or files (e.g. in case of a bug, a log file could also be provided).

[1] http://trac.edgewall.org/

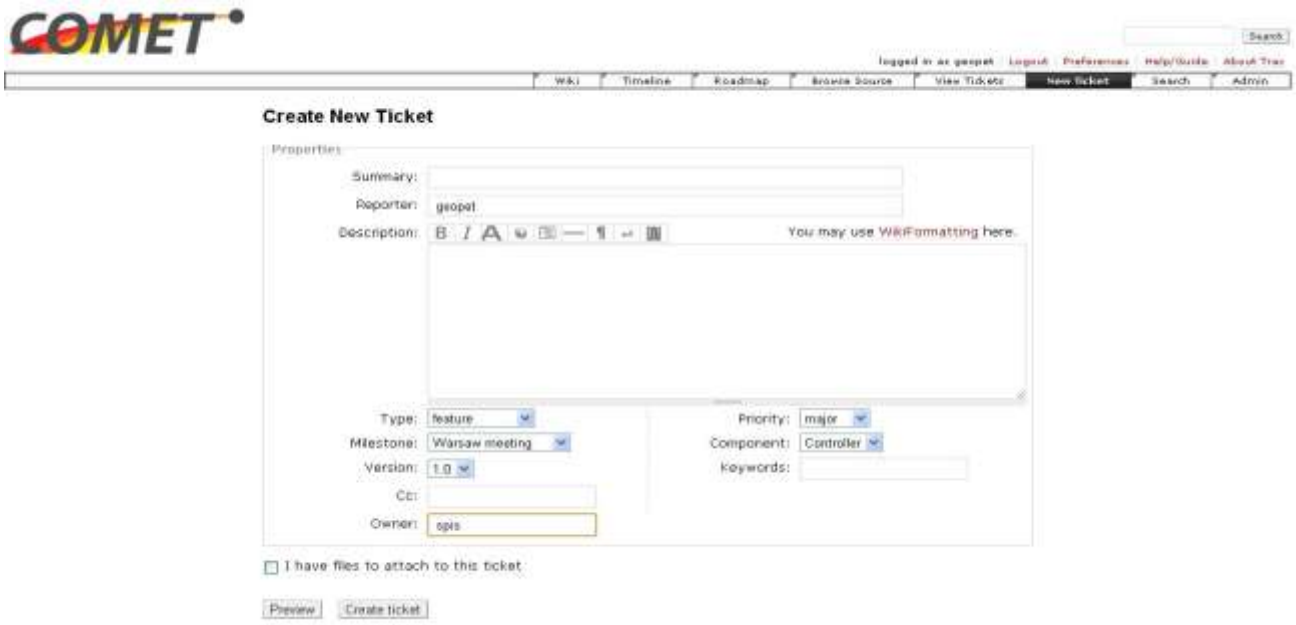

Figure 2: New ticket creation in TRAC

- **Accepted:** Upon creating a new ticket and assigning it to a developer, the developer must accept the ticket and start developing.
- **Resolved:** Upon completing the implementation and resolving the issue, the responsible developer should send a component release to the integrator, change the status of the ticket to resolved and wait for feedback from the integrator, who will check if the system works properly.
- **Closed:** When an issue is resolved and everything works as expected, the integrator should change the status of the ticket to closed. In any other case, the ticket is still considered as open. If any issues related to this ticket occur in the future, the integrator may reopen the ticket and assign it to a developer again.
-

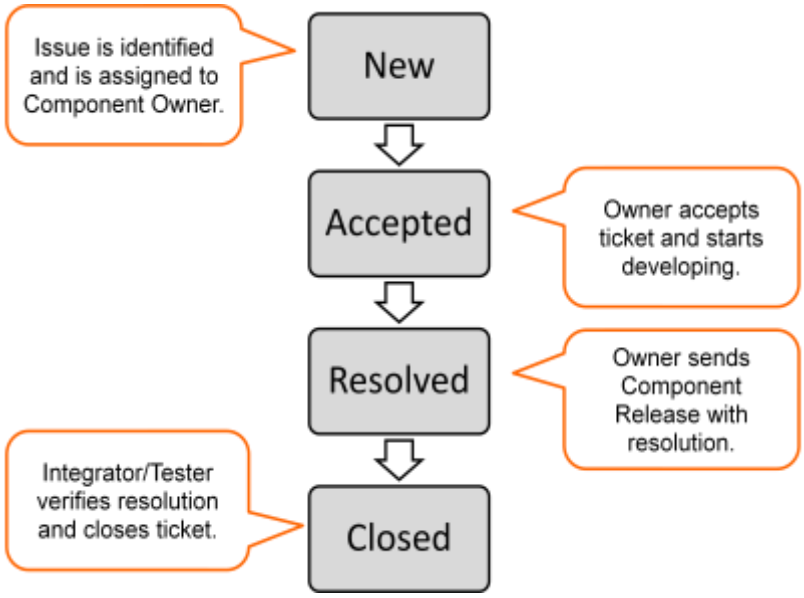

Figure 3: Ticket Lifecycle

3.2.1.6 Continuous integration server

Hudson[2] is used as the continuous integration server of the COMET decupled prototype, and more specifically to automatically build and test the CME components. The Hudson server is configured to poll every hour the SVN system for changes. Once a change is detected, the build process is triggered. The outcome of the build process is announced to the users with administrative rights as well as to the users whose changes triggered the build. This way in case of an error, the respective developers are notified in time to fix it. Apart from this hourly triggered process, Hudson is configured to build the project every midnight as well.

3.2.2 Integration testbeds

For the validation of the COMET decoupled prototype, several validation testbeds were defined prior to particular system release. Each testbed was designed and deployed based on the requirements of the planned integration tests and the hardware and software needs of the developed entities. The main testbeds designed and implemented excluding minor variations are the testbeds produced for releases 1.0, 2.0 and 3.0. The integration testbeds were developed on Virtual Machines (VMs). The ESXi was the virtualization tool used to deploy VMs on a set of validation servers. Each entity was installed on a single VM, where required software and hardware resource were assigned. This approach was applied during the validation of all system releases. The selection of operating systems, hardware resources and the associated software are summarised in Table 3.

Table 3: Resource allocation for integration testbeds

Entity	OS	Apps	RAM	HDD
CME	Windows Server	Apache Ant, Java Development Kit, .NET, MySQL Server	256MB	5GB
CRE	Windows Server	Apache Ant, Java Development Kit	256MB	5GB
RAE	Ubuntu	Apache Ant, Java Development Kit, sqlite3, waf	256MB	5GB
SA	Ubuntu	Apache ANT, Java Development Kit, MySQL	256MB	5GB
CAFE forward/edge	Tiny CORE	Quagga (Zebra, OSPF), Python	128MB	0.5GB
CS-SMA	Windows Server	Apache ANT, Apache Tomcat, Java Development Kit, MySQL, VLC	256MB	5GB
CP	Windows	Apache ANT, Java Development Kit, MySQL	256MB	5GB
CC	Windows	VLC, Mozilla Forefox	256MB	5GB

3.2.2.1 Testbed for system release v1.0

This was the first testbed developed in order to satisfy the first validation tests. It involved 7 VMs located in a single domain. Thus, the testbed was implemented purely on a virtual environment without any physical networking devices as applied to next testbeds described in sections 3.2.2.2 and 3.2.2.3.

[2] http://java.net/projects/hudson/

The VMs developed were applied to the following entities:

1. Domain Router
2. Content Client
3. CRE along with its Content Registration Server (web server)
4. CME
5. RAE
6. Content server
7. Content provider

The testbed used for the validation tests relevant to the first release of the COMET decoupled prototype is shown in Figure 4.

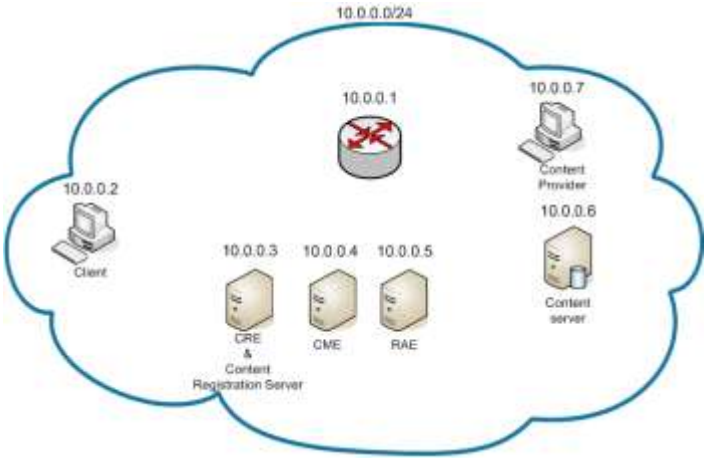

Figure 4: Integration testbed for system release v1

3.2.2.2 Testbed for system release v2.0

The testbed for the second release was developed in order to satisfy the testing of the partially functional and integrated COMET system. The functionalities of path configuration, content forwarding and server awareness were not integrated with the rest of the system. Thus, v2 testbed was composed of 3 domain networks connected by physical network devices. Each domain was implemented on individual virtual environments. No access/sub-domain networks were developed, due to the absence of the network layer functionalities (CAFE). The testbed used for all validation tests relevant to the second COMET system release is shown in Figure 5.

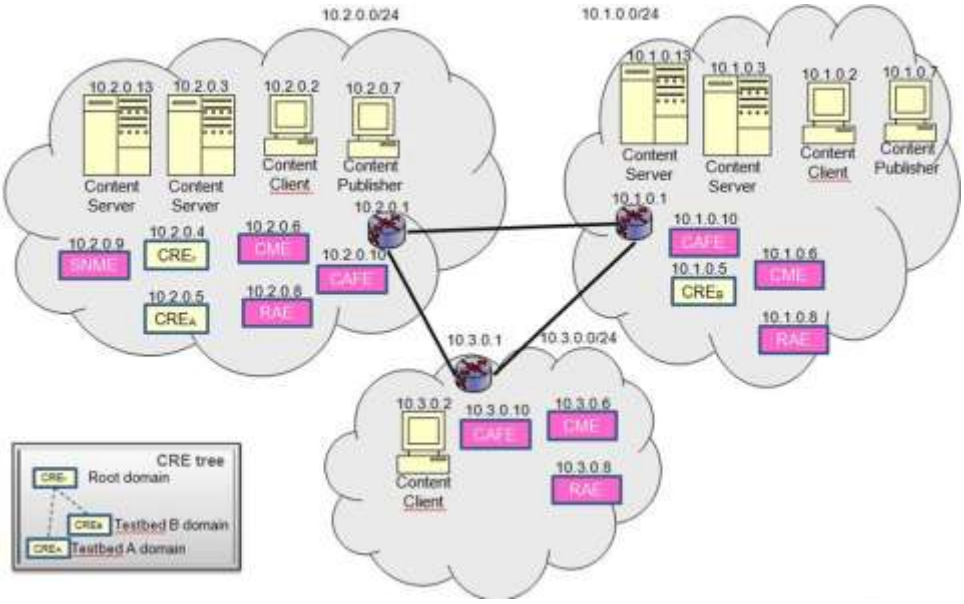

Figure 5: Integration testbed for system release v2

3.2.2.3 Testbed for system release v3.0

The final integration testbed was composed of 3 domain networks, 5 access networks and 32 entities. This enabled full functional and overall tests of the COMET decoupled prototype. Virtual switching was applied to all virtual LAN networks enabling full switching capability to the sub/domain networks. However, the inter or intra domain IP routing was performed via the use of physical routers as also performed in the previous release. The final testbed used for all final validation tests relevant to the final COMET system release is shown in Figure 6.

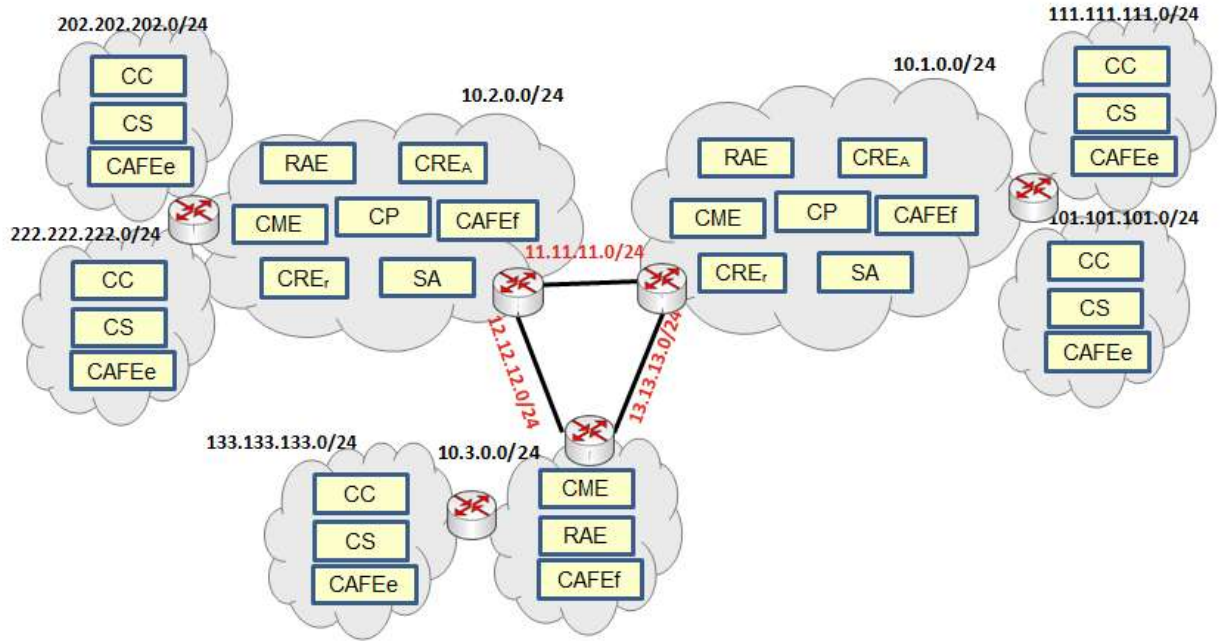

Figure 6: Integration testbed for system release v3

3.2.3 Integration procedures

For the efficient and successful implementation of the COMET prototype, the partners used the described above implementation and integration framework, as well as followed a common procedure (Figure 7):

1. The consortium decided on the desired features of the next release of the software periodically (every plenary meeting),
2. Mechanisms and algorithms for the added features were specified by the responsible partners,
3. Development of components by the responsible partners (assignment of components to partners and agreement on the common implementation and integration framework was done prior to initialization of implementation),
4. Integration of developed components and basic sanity tests,
5. Validation of software through system functionalities' testing in the validation testbed, and
6. System release

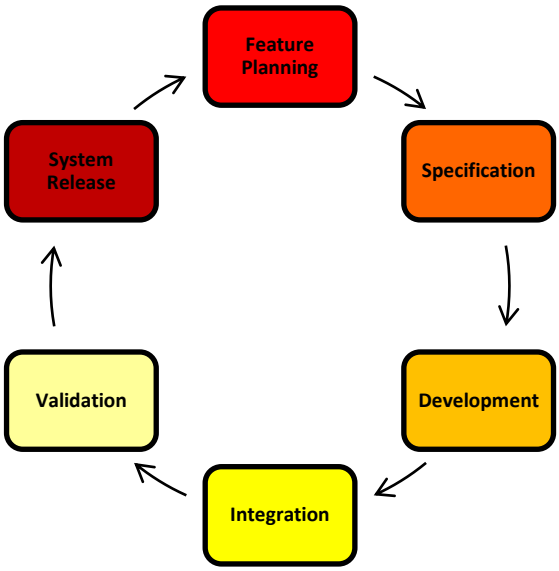

Figure 7: System release iterative process

Development of components is an iterative process as well, in which responsible developers implement their components and perform some self-contained (unit) tests to validate their code. In case of tests' failure, developers restart the implementation of their components, until these tests become successful. Then, they release their components and notify the integration team of COMET.

Upon components release, integration process starts aiming to perform simple tests among implemented components. Found bugs and issues are reported to the issue tracking platform (TRAC) and developers get automatically notified to resolve the issues. Finally, when all issues are resolved, system tests are performed in the validation testbed, and all issues found are reported to be resolved in next releases. This iterative process ensures the efficiency and success of the implementation tasks, as well as reduces the possibility of errors and conflicts.

The COMET project released 3 main versions of its software, enhancing new features and operations in the software, and 2 additional ones, mainly resolving bugs and discovered issues from previous releases. The history of releases is presented in Section 3.3 below.

3.3 History of system releases

In this chapter, we present three releases developed for COMET decoupled prototype. For each system release, we briefly introduce the implemented features, the scope of validation tests as well as the identified problems, bugs and open issues. The final validation tests, results and conclusions are presented in chapter 3.4. The detailed information about validation tests performed for each system release included in Appendixes A, B, and C.

3.3.1 Release v1.0

Version 1.0 was the first release of COMET decupled prototype, introducing certain basic features and interfaces. Its aim was to implement:

- the name resolution system within COMET (CREs hierarchy, CP web application)
- the basic structure and functionalities of key CMP entities (CC, CME and CS)
- the routing awareness system (RAEs) and RAE-CME interface

The features implemented in release 1.0 are summarised in Table 4.

Table 4: The features implemented in release 1.0

Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#4	CC request handler	CC	closed	fixed	v 1.0	Feature	major	ruben	06/03/11
#12	CC direct request	CC	closed	fixed	v 1.0	Feature	major	ruben	06/03/11
#5	CC-CME interface	CC	closed	fixed	v 1.0	Feature	major	ruben	06/03/11
#3	Content Publication	CP	closed	fixed	v 1.0	Feature	major	geopet	05/27/11
#1	CME-CC interface	CME	closed	fixed	v 1.0	Feature	major	souse	05/03/11
#7	Content Name resolution	CME	closed	fixed	v 1.0	Feature	major	geopet	05/03/11
#2	CRE setup and run	CRE	closed	fixed	v 1.0	Feature	major	geopet	05/03/11
#9	RAE-CME interface	RAE	closed	fixed	v 1.0	Feature	major	jarek	04/20/11
#10	CME-RAE interface	CME	closed	fixed	v 1.0	Feature	major	souse	04/20/11
#6	Mediation controller	CME	closed	fixed	v 1.0	Feature	major	geopet	04/18/11
#14	CME administration	CME	closed	fixed	v 1.0	Feature	major	geopet	04/18/11
#8	Decision maker	CME	closed	fixed	v 1.0	Feature	major	geopet	04/18/11
#13	CME database	CME	closed	fixed	v 1.0	Feature	major	souse	04/18/11

The validation of release 1.0 aimed to verify basic functionality related to content publication, content resolution and routing awareness. The validation tests were performed in the validation testbed presented in Figure 4. Validation tests of were performed in three groups of tests that are:

1. Content publication
   - Content Publication with correct and incorrect credentials
   - Registration of new content
   - Modification of content source
   - Modification of content server
   - Verification that new content server was added to handle new content
2. Content resolution
   - Client requests existing content
   - Client requests non-existing content
3. Routing awareness
   - Add routing prefix
   - Remove routing prefix
   - Update information about paths
   - Update information about domain provisioning
   - Reset/unavailability of RAE
   - Updating large number of prefixes.

Most of the tests have passed, but some issues were found during validation. They are presented in Table 5.

Table 5: Open issues found in validation of release 1.0

Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#16	When putting invalid input in "Number of Traffic Descriptors" server error appears	CP	accepted		v 1.1	Bug	major	geopet	05/31/11
#18	Create content with same name as existing one throws an exception	CP	accepted		v 1.1	Feature	major	geopet	06/02/11
#15	Session timesout very quickly	CP	accepted		v 1.1	Bug	minor	geopet	05/31/11
#17	COMET/admin and COMET/ADMIN should be same	CP	accepted		v 1.1	Bug	minor	geopet	06/02/11

3.3.2 Release v1.1

Version 1.1 was a supporting release of COMET decupled prototype, aiming to fix certain bugs and issues found in v1.0. The main issues from v1.0 were:

- Lack of self-contained tests in most implemented entities
- Bugs in CP web application

Table 6: The features implemented in release 1.1

#18	Create content with same name as existing one throws an exception	CP	closed	worksforme	v 1.1	Feature	major	geopet	06/21/11
#19	CC code modifications	CC	closed	fixed	v 1.1	Bug	major	ruben	06/21/11
#20	RAE code modifications	RAE	closed	fixed	v 1.1	Bug	major	jarek	06/20/11
#22	CRE code modifications	CRE	closed	fixed	v 1.1	Feature	major	geopet	06/14/11
#17	COMET/admin and COMET/ADMIN should be same	CP	closed	fixed	v 1.1	Bug	minor	geopet	06/14/11
#15	Session timeout very quickly	CP	closed	fixed	v 1.1	Bug	minor	geopet	06/14/11
#16	When putting invalid input in "Number of Traffic Descriptors" server error appears	CP	closed	fixed	v 1.1	Bug	major	geopet	06/14/11
#21	CP code modifications	CP	closed	fixed	v 1.1	Feature	major	geopet	06/14/11

The validation tests defined for v1.0, were repeated for v1.1 aimed to test missing test scenarios and found bugs from v1.0. After validation of v1.1, no particular bugs or issues were reported.

3.3.3 Release v2.0

Release 2.0 enhanced core COMET functionalities in the COMET decupled prototype

- Routing awareness in CME (CME-RAE interface)
- Server awareness (CS monitoring, CS-SNME interface)
- Decision algorithm
- Path discovery, provisioning and configuration (inter-CME interface)
- Minor features added to existing entities and components

The features implemented in release 2.0 are summarised in Table 7

Table 7: The features implemented in release 2.0

v2.0 components release

Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#35	CS-SNME interface	CS	closed	fixed	v 2.0	Feature	major	ruben	11/25/11
#36	CS-SNME interface (Server awareness)	SNME	closed	fixed	v 2.0	Feature	major	ruben	11/23/11
#37	Server load DB	SNME	closed	fixed	v 2.0	Feature	major	ruben	11/23/11
#28	Path Configuration	CME	closed	fixed	v 2.0	Feature	major	geopet	11/23/11
#33	Server monitoring agent	CS	closed	fixed	v 2.0	Feature	major	ruben	11/23/11
#34	New CS type	CS	closed	fixed	v 2.0	Feature	major	ruben	11/23/11
#39	New CC type	CC	closed	fixed	v 2.0	Feature	major	ruben	11/23/11
#26	Decision making	CME	closed	fixed	v 2.0	Feature	major	geopet	11/10/11
#29	Path Provisioning	CME	closed	fixed	v 2.0	Feature	major	geopet	11/03/11
#25	inter-CME protocol	CME	closed	fixed	v 2.0	Feature	major	geopet	11/03/11
#27	Path Discovery	CME	closed	fixed	v 2.0	Feature	major	geopet	11/03/11
#24	CME database	CME	closed	fixed	v 2.0	Feature	major	geopet	11/03/11
#30	CME admin new features	CME	closed	fixed	v 2.0	Feature	major	geopet	11/03/11
#31	Content Publisher modifications	CP	closed	fixed	v 2.0	Feature	major	geopet	11/03/11
#32	CRE modifications	CRE	closed	fixed	v 2.0	Feature	major	geopet	11/03/11
#11	Path storage	CME	closed	fixed	v 2.0	Feature	major	geopet	11/03/11
#9	RAE-CME interface	RAE	closed	fixed	v 2.0	Feature	major	jarek	11/03/11
#38	Route Ranking algorithm	RAE	closed	fixed	v 2.0	Feature	major	jarek	11/03/11

The validation of release 2.0 was performed to verify new features. It was performed in the validation testbed v2.0 presented in Figure 5. Validation tests were focused on:

1. Content publication
   - Publish, alter/edit content from domain 1 or 2
   - Delete content created in all domains
   - Batch update/edit or delete of content

2. Content resolution
   - Resolution of local content
   - Resolution of remote content
   - Non-static content
   - Decision process:
     - CoS
     - path length
     - QoS parameters: IPTD, IPLR and BW

Open issues reported after v2.0 tests are presented in Table 8.

Table 8: Open issues found in validation of release 2.0

Ticket	Summary	Type	Status	Priority	Milestone	Component
#54	Bug in Server Awareness when closing multiple servers	Bug	closed	critical	v3.0 components release	SNME
#50	Incorrect strict mode for BW	Bug	closed	major	v3.0 components release	CME
#52	Check for already added content servers	Bug	new	minor	v3.0 components release	SNME

After validation of release 2.0, three bugs were identified. They were fixed in the release 3.0.

3.3.4 Release v3.0

Version 3.0 introduced new features to the software and is the final IPv4-only compatible release of the COMET decupled prototype:

- Server awareness in CME (SNME-CME interface, changes in decision algorithm)
- CAFE configuration in CME (CME-CAFE interface)
- Content forwarding through the CAFEs
- Browser-based CC

Table 9: The features implemented in release 3.0

v3.0 components release

Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#52	Check for already added content servers	SNME	closed	invalid	v 3.0	Bug	minor	ruben	01/10/12
#56	Multiple streams identification for same (cc, cs)	CME	closed	fixed	v 3.0	Bug	major	geopet	01/10/12
#50	Visual CC	CC	closed	fixed	v 3.0	Feature	major	ruben	01/25/12
#54	Bug in Server Awareness when closing multiple servers	SNME	closed	fixed	v 3.0	Bug	critical	ruben	01/25/12
#53	SNME JUnit tests	SNME	closed	fixed	v 3.0	Support	major	ruben	01/24/12
#43	SNME-CME interface	SNME	closed	fixed	v 3.0	Feature	major	ruben	01/18/12
#44	inter-CME protocol update	CME	closed	fixed	v 3.0	Feature	major	geopet	01/10/12
#42	CME-SNME interface	CME	closed	fixed	v 3.0	Feature	major	geopet	01/04/12
#46	Modification to Decision Maker	CME	closed	fixed	v 3.0	Feature	major	geopet	01/04/12
#40	CME-CAFE interface	CME	closed	fixed	v 3.0	Feature	major	geopet	01/04/12
#48	Server-awareness	CME	closed	fixed	v 3.0	Feature	major	geopet	01/04/12
#47	CME db and admin interface update	CME	closed	fixed	v 3.0	Feature	major	geopet	01/04/12
#45	Resources management	CME	closed	fixed	v 3.0	Feature	major	geopet	01/04/12
#50	Incorrect strict mode for BW	CME	closed	fixed	v 3.0	Bug	major	geopet	01/04/12
#49	Content forwarding	CAFE	closed	fixed	v 3.0	Feature	major	jarek	12/21/11
#41	CAFE-CME interface	CAFE	closed	fixed	v 3.0	Feature	major	jarek	12/21/11

The validation of release 3.0 was performed to verify content resolution and consumption. It was performed in the validation testbed v3.0 presented in Figure 6. Validation tests were focused on:

1. Fixing bugs identified in release 2.0
2. Content resolution and consumption
   - Inter Domain CME path preparation
   - Inter Domain CME BW management

No particular issues were reported for v3.0.

3.3.5 Release v3.1

Version 3.1 was a supporting release of COMET software aiming to:

1. Verify IPv6 compatibility of developed modules
2. Integration of visualisation tools developed for demonstration of SNME and CAFE
3. Verify CoS handling in CAFE.

Table 10: The features implemented in release 3.1

Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#65	QoS profiles in CP publication	CP	closed	fixed	v 3.1	Feature	major	geopet	06/26/12
#61	IPv6 compatibility	CRE	closed	fixed	v 3.1	Feature	major	geopet	06/26/12
#60	IPv6 compatibility	CP	closed	fixed	v 3.1	Feature	major	geopet	06/26/12
#59	IPv6 compatibility	CME	closed	fixed	v 3.1	Feature	major	geopet	06/26/12
#57	IPv6 compatibility	CAFE	closed	fixed	v 3.1	Feature	major	andrzej	06/26/12
#66	CoS handling in CAFE	CAFE	closed	fixed	v 3.1	Feature	major	andrzej	06/22/12
#64	IPv6 compatibility	SNME	closed	fixed	v 3.1	Feature	major	ruben	06/20/12
#68	CAFE graphic tool	CAFE	closed	fixed	v 3.1	Feature	major	andrzej	06/18/12
#63	IPv6 compatibility	RAE	closed	fixed	v 3.1	Feature	major	andrzej	06/18/12
#58	IPv6 compatibility	CC	closed	fixed	v 3.1	Feature	major	ruben	06/15/12
#62	IPv6 compatibility	CS	closed	fixed	v 3.1	Feature	major	ruben	06/15/12
#67	SNME graphic tool	SNME	closed	fixed	v 3.1	Feature	major	ruben	06/15/12

The validation of IPv6 compatibility was finished for CC, CAFE, CS, CP, CRE, RAE and SNME and considered applications. The obtained results confirmed that content could be delivered on IPv6. However, there are some minor validation tests pending after closing this deliverable. These results and the other issues, if detected during performance tests, will be reported in amendment.

3.4 Validation of functionalities of decoupled entities

In this chapter we present tests performed to validate the COMET decoupled prototype. Our primary objective was to verify functionality of particular COMET entities and their capabilities for cooperation. We followed the bottom-up approach, so we first validate the sub-processes and then we focus on overall validation. In particular, we focused on major sub-processes that are: (1) content publication, (2) name resolution, (3) server awareness, (4) routing awareness, (5) decision algorithm and path discovery, (6) path configuration, and (7) content delivery.

3.4.1 Content publication

The tests were performed initially on the release 1.0 and finalised prior to the release 2.0. The details of test procedures together with results obtained prior to the first release can be found in the Annex 10.1.

3.4.1.1 Objective

The objective of tests related to Content Publication is to verify: 1) the CRE and 2) the interface between CP and CRE. The following test cases have been defined:

- Test case 1: Content registration,
- Test case 2 :Content update/edit,
- Test case 3: Content removal,
- Test case 4: Content batch publication.

The test cases 1-3 correspond to the content publication via CP administration web page, whereas test case 4 focuses on publication of content in batches.

The aim of test cases is to confirm the proper content storage and registration as described in D3.2 [2]. This effectively refers to the interface between the CP and the CRE together with the sub-processes of content registration, update/edit, delete and batch publication. These processes were expected to be functional when provided correct data and also restrict erroneous data logging and prevent false data provided by the publisher.

3.4.1.2 Test case 1: Content registration

The objective of this test case is the validation of content registration in the CRE instantiated by CP. For this purpose, we consider both positive (correct values of fields of CR) and negative (incorrect values of fields of CR) scenarios.

Several records were created in order to verify that the CP could publish/create a record at the local CREs. The main steps of publishing performed were as shown in the below example:

Publish content *domain_1_video_rtsp* and *domain_1_video_http*

1. Access Content Publisher by accessing the administration page at *http://localhost:8090/Publisher/index.jsf*
2. Enter the correct credentials.
3. Register domain_1_video_http at COMET1 namespace with the following details.
    - Content Name: *COMET1/domain_1_video_http*
      Number of Content Sources: 1
      MIME Type (Content Type): mp4
      CoS: BTBE
      Traffic descriptor: Peak Bit Rate rate 1000 kbps
      IPTD[3]: 0.1

[3] Value irrelevant for this section of testing

IPLR[3]: 0.1
Priority: 1
Application protocol: HTTP
Transport protocol: TCP
Port: 8080

Number of Content Servers: 1
IP address: 111.111.111.3
Path: /COMET/domain_1_video_http.mp4
CME IP: 10.1.0.6

4. Verify after creation that record exists at main screen.
5. Register *domain_1_video_http* with the following details.

- Content Name: *COMET1/domain_1_video_rtsp*
Number of Content Sources: 1
MIME Type (Content Type): mp4
CoS: PR
Traffic descriptor: Peak Bit Rate rate 1000 kbps
IPTD[3]: 0.1
IPLR[3]: 0.1
Priority: 1
Application protocol: RTSP
Transport protocol: TCP
Port: 5544

Number of Content Servers: 1
IP address: 101.101.101.3
Path: /COMET/domain_1_video_rtsp.mp4
CME IP: 10.1.0.6

6. Verify after creation that record exists at main screen.

Delete content *domain_1_video_rtsp* and *domain_1_video_http* and repeat the above procedure with false/unreal values in the text fields. E.g. IP Packet Loss Ratio: 1.1 and Transport port: 66000

3.4.1.3 Test case 2: Content update/edit

The process of updating/editing content records was validated by performing several alterations to existing records, e.g. updating *domain_1_video_rtsp*:

1. Access Content Publisher at IP 10.1.0.7 by accessing the administration page at *http://localhost:8090/Publisher/index.jsf*
2. Enter the correct credentials.
3. Select *domain_1_video_rtsp* by drop down window
4. Make the following alterations:

CoS: PR
Traffic descriptor: Peak Bit Rate rate 100 kbps

5. Preview the record and then update.

6. Select the record from main screen and verify that it has been updated[4].

3.4.1.4 Test case 3: Delete content

The functionality of deleting records in CRE was validated by deleting content records. The testing procedure was as follows:

1. Repeat steps 1-3 in previous section.
2. Delete record.
3. Verify that the record is not present in CRE database.

3.4.1.5 Test case 4: Content batch publication

The objective of this test is to validate the content publication in batches. All operations described on test cases 1-3 could be also performed in batches. Thus, the tests performed via the CP administration web page were repeated using batch text files.

- Content registration (positive scenario)

Using a batch file, publish content *domain_1_video_rtsp* and *domain_1_video_http* available from CS 10.1.0.3:

1. Create a batch text file and add the following instructions:

   ```
   CREATE COMET1/domain_1_video_http
   CONTENT_SOURCE mp4 BTBE 1000 1 1 1 http tcp 8080 1 10.1.0.3 /COMET/domain_1_video_http.mp4 10.1.0.6
   CREATE COMET1/domain_1_video_rtsp
   CONTENT_SOURCE mp4 BTBE 1000 1 1 1 rtsptcp 5544 1 10.1.0.3 /COMET/domain_1_video_rtsp.mp4 10.1.0.6
   ```

2. Access Content Publisher at IP 10.1.0.7 by accessing the administration page at http://localhost:8090/Publisher/index.jsf
3. Enter the correct credentials and select 'Upload your batch file'.
4. Upload batch file created in step 1
5. Verify that the content records were created and all fields contain the correct values.

- Content update (positive scenario)

Using a batch file, edit content *domain_1_video_rtsp*:

1. Create a batch text file and add the following instructions:

   ```
   EDIT COMET1/domain_1_video_rtsp
   CONTENT_SOURCE mp4 PR100 1 1 1 rtsptcp 5544 1 10.1.0.3 /COMET/domain_1_video_rtsp.mp4 10.1.0.6
   ```

2. Repeat steps 2-4 from previous test.
3. Verify that the content record was updated to the new values.

- Content removal (positive scenario)

[4] CP administration web page provides access to the records stored in CRE. No local records (i.e. in CP) are available in case of disconnection with CRE.

Using a batch file, delete created content:

1. Create a batch text file and add the following instructions:

   ```
   DELETE COMET1/domain_1_video_rtsp
   DELETE COMET1/domain_1_video_http
   ```

2. Repeat steps 2-4 from first test.
3. Verify that the content records were deleted.

- Content registration (negative scenario)

Using a batch file, publish content using false commands and non-existing content. E.g. using the following batch file:

```
CREATE COMET1/domain_1_video_http
CONTENT_SOURCE mp4 BTBE 1000 1 1 1 http tcccp 8080 1 10.1.0.3 /COMET/domain_1_video_http.mp4 10.1.0.6
EDIT COMET1/JOHN_DOE
CONTENT_SOURCE mp4 PR100 1 1 1 rtsptcp 5544 1 10.1.0.3 /COMET/domain_1_video_rtsp.mp4 10.1.0.6
```

3.4.1.6 Results

The functionalities of publication performed as expected. The CP was capable to perform remotely all alterations required to the CRE database restricting any erroneous data caused by the user. The results and further detail of the above test procedures can be found in the Annex 11.1.

3.4.2 Name resolution

The integrity of the CC, CME and CRE was finalized prior to the first release and all the necessary validations were performed using v1 testbed.

3.4.2.1 Objective

The objective of this test is to validate name resolution process.

The client had to be capable to send requests for content to the CME and the resolver component then locate the requested content at the appropriate authoritative CRE. Thus, the CC-CME interface, CME-CRE interface and the resolver were expected to interact as described in D3.2 [2] and successfully return/resolve an existing content. The details of the following test procedures can be found in the Annex 10.2.

3.4.2.2 Test case 1: Request existing content

Several contents were published at different domains dispersed at different authoritative CREs using v1 testbed. The main steps of the test procedure were as follows:

1. Confirm CME 10.5.90.244 server is running by locating the equivalent Java message.
2. Login into CME web interface.
3. Select ‘configure CRE’ and provide ip of CRE 10.5.90.245 and port to what CRE is listening (i.e. 2641).
4. Launch Client.exe and provide remote IP adderss of cme 10.5.90.244, remote port: 9091 and existing content name. COMET/BIRTHDAY_2010_10235

3.4.2.3 Test case 2: Request non-existing content

The test confirmed that similar naming even at existing domain would still not return a record. The above steps were repeated with content name COMET/BIRTHDAY.

3.4.2.4 Results

CME resolver collected the appropriate content requested by the CC as verified by the CME output.dat file. Furthermore, when requesting non-existing content, CME received appropriate message from authoritative CRE and responded appropriately to CC. The details of the above test procedures and equivalent results can be found in the Annex 10.2.

3.4.3 Server awareness

Server awareness was tested at two stages due to the integrity of the COMET decupled prototype at each release. Prior to release 2.0 excluding CME and prior to release 3.0 including CME.

3.4.3.1 Objective

The objective is to validate the CME-CME interface, CME-SNME interface, SNME-SMA interface and the correct calculation of server status by the SMA based on memory, bandwidth and processor availability.

3.4.3.2 Test case 1: SNME-SMA (isolated test)

The tests involved one SNME and several CSes (SMAs), excluding the CME interface and CME in general. All the responds H, M, L, U and D transmitted by the SMA to the SNME were verified. The following test procedure was applied to all three SNMEs i.e. to each SNME allocated at each domain/ISP. The results of the following test procedures can be found in the Annex 12.4.

1. Access the domain CSes and initialise their SMA.
2. Record the levels for H, M and L from the configuration file.
3. Run a software application that will require the necessary amount of resources (i.e. bandwidth or memory or processing). E.g. play a movie file for HDD, download a large file at high speed, run many apps to load memory etc.
4. Access SNME monitoring the CS.
5. Confirm the SNME is receiving a status from the SMA using the information provided in the SMA and SNME terminal.
6. Access the SQL table CS_STATUS_INFO from SNME and record the status for the CS.
7. Verify that the status of the server is the appropriate one.
8. Repeat steps 3-7, by running a necessary number of different or same applications (e.g. play 2 movies concurrently) necessary to achieve the required status of L ,M and H.
9. Access SMA and alter the threshold levels and the margin.
10. Repeat steps 3-7 in order to verify that the status are the appropriate, based on the new levels and margin.
11. Disconnect server from network as described in step 6 verify that the status goes from U to D. (negative scenario).

3.4.3.3 Test case 2: CME-SNME-SMA (integrated test)

The tests confirmed that CME could collect the real-time status from any SNME (i.e. local or remote) for several CSes using the inter-CME interface (i.e. when remote SNME). The tests validated the proper integration with CME and Server Awareness functionality in general. The testing procedure was repeated appropriately in order to validate/include all SNMEs and the results can be found in the Annex 12.4

1. Access and initialize agents at CS 111.111.111.3 and 101.101.101.3.
2. Record the levels for H, M and L from the configuration file.

3. Run a software application that will require the necessary amount of resources (i.e. bandwidth or memory or processing). E.g. play a movie file.
4. Access SNME at 10.1.0.9.
5. Confirm that the SNME 10.1.0.9 is receiving the appropriate status.
6. Repeat steps 3-5, by running a necessary number of different or same applications (e.g. play 2 movies concurrently) necessary to achieve different status on different CS.
7. Access CME 10.1.0.6 and run terminal on debug mode.
8. Perform consumption as described in previous sections from CC 111.111.111.2.
9. For each consumption verify that the status received in CME terminal and SNME are identical and match the expected status by performing step 3.
10. Repeat all steps performing consumptions from remote CCs for the same content e.g. using CC 222.222.222.2.

3.4.3.4 Results

During the first stage of testing (i.e. isolated) the status was calculated correctly and the unidirectional messaging from SMA to SNME was appropriately stored in the SNME database. However the status of CS was not updating correctly and the CS status remained idle. Thus, a fix was applied and the SNME entity was re-validated during integrated tests. Furthermore, all tests performed in the second stage of testing were successful and CME could collect at any point the status of the SNME. The results of the isolated and integrated tests can be found in the Annex sections 11.3 and 12.4 consecutively.

3.4.4 Routing awareness

The RAE was tested prior to the first release. The tests required one CME and several RAEs .

3.4.4.1 Objective

Confirm that the RAE entity could calculate routing paths and update information to path storage component of CME using the CME-RAE interface. The results of the following tests can be found in the Annex 10.3.

3.4.4.2 Test case 1: Simple configuration (CME-RAE)

In this test, seven domains were formed i.e. creating a tiered topology. Each domain runs RAE with 4 prefixes. We change the number of available prefixes and domains just by alteration of RAE configuration file and then restarting the entity. Thus, every alteration was updated in the CME and the rest of the RAE. The alterations performed were editing/updating of path information and addition/deletion of prefixes.

3.4.4.3 Test case 2: Stress test (RAE-RAE)

This test involved a pair of domains only. One domain hosted a variable number of prefixes, where the default count was 10000 and the other domain received the paths and propagated them to the CME. Thus, the RAE entity and mainly RAE-RAE interface had to cope with the update of large number of prefixes.

3.4.4.4 Results

Tests completed successfully even during stress tests confirming appropriate functionality and messaging between the entities of CME-RAE and RAE-RAE. The results can be found in the Annex 10.3.

3.4.5 Decision process and path discovery

3.4.5.1 Objective

Validate all the parameters influencing the selection of path in order to confirm that appropriate path and server were selected based on user, path and content restrictions/parameters i.e. CoS, Path length, IPTD, IPLR, BW and server load. The results of the tests described in this section can be found in the Annex 11.2 and 12.2

3.4.5.2 Test case 1: Classes of Services

Three CoSs were examined individually in all 3 domains. The below procedure demonstrates the steps followed in order to perform a content request by the CC 111.111.111.2 with CoS BTBE[5]:

1. Create a content for domain 1 with CoS BTBE.
2. Access CME administration at IP 10.2.0.6 by accessing the administration page at *http://localhost:8090/CME/main.jsf*
3. Alter CoS of CC 111.111.111.2 at CME administration to BTBE.
4. Access CC at 111.111.111.2
5. Request the content created in step 1 as follows:

   Client 10.2.0.6 9091 {content name}
6. If requested, provide associated application for the content's application protocol.
7. Verify the selected handling application is initialized e.g. VLC, and the correct content is streaming.
8. Verify from edge CME terminal that the generated path keys do not contain PR paths.

3.4.5.3 Test case 2: Path length

The testes were applied using v2 testbed and equivalently the path length could not be more than 5 hops and the available paths could not be more than two. Thus, the tests confirmed when reservation levels for path length was '5', we had two candidate paths and when '4' we had just one candidate path. The test was performed to all CMEs, e.g.:

1. Access CME administration at IP 10.3.0.6 by accessing the administration page at *http://localhost:8090/CME/main.jsf*
2. Alter path length reservation level to 3 and aspiration to 0.5 under the *decision process* tab.
3. Alter type to strict for path length, all other parameters should be tolerant.
4. Access CC at 133.133.133.2
5. Request content available in domain 1.
6. Access CME terminal[6] and confirm that there are two candidates and the selected path has the smallest length (i.e. 2).
7. Verify the selected handling application is initialized e.g. VLC, and the correct content is streaming.
8. Repeat steps 1-4 with reservation level two.

[5]The content is available with PR and BTBE paths. BE content is not mediated and effectively path selection is unnecessary.

[6] CME terminal provided debug information e.g. ranking paths and values.

9. Access CME terminal and confirm that there is only one candidate and the selected path has length 2.
10. Verify the selected handling application is initialized e.g. VLC, and the correct content is streaming.

3.4.5.4 Test case 3: Values of IPTD, IPLR

These two parameters were depended solely by the content equivalent requirements as specified by the publisher. Thus, the content had to be delivered only under the necessary conditions. The following test was performed on both parameters when performing inter-domain consumptions.

1. Access RAE at IP 10.3.0.8.
2. Initialise a terminal at RAE's directory and extract the paths from the database as follows:

   *$ sqlite3 --header db3.sqlite3 "select * from paths;"*
3. Record the IPTD and the IPLR for the expected two paths.
4. Access Content Publisher at IP 10.1.0.7 by accessing the administration page at *http://localhost:8090/Publisher/index.jsf*
5. Enter the correct credentials.
6. Alter *domain_1_video_http* existing at *COMET1* namespace with the following details.

   IPTD: {minimum delay recorded in step 3}
7. Verify that alterations were saved.
8. Access CME administration at IP 10.3.0.6 by accessing the administration page at *http://localhost:8090/CME/main.jsf*
9. Alter type to strict for IPTD, all other parameters should be tolerant.
10. Access CC at 10.3.0.2
11. Request content available in domain 1.
12. Access CME terminal and confirm that there is only one candidate i.e. the path with minimum delay.
13. Verify the selected handling application is initialized e.g. VLC, and the correct content is streaming.
14. Repeat step 4-11, but using the maximum delay and an aspiration to any value less than 1 and above 0.
15. Access CME terminal and confirm that there are two candidates and the selected path is still the path with minimum delay.
16. Verify the selected handling application is initialized e.g. VLC, and the correct content is streaming.

3.4.5.5 Test case 4: Server load

The server can only have reservation level 3 and effectively the maximum state is busy. The test procedure applied to the parameters of aspiration and type was as follows:

1. Access the SQL table CS_STATUS_INFO from SNME and record the status for the expected CS.
2. Access CME administration at IP 10.3.0.6 by accessing the administration page at *http://localhost:8090/CME/main.jsf*

3. Alter type to tolerant for server load, all other parameters should be tolerant.
4. Access CC at 10.3.0.2
5. Request content available only from one CS that currently is deactivated.
6. Access CME terminal and confirm that there is only one candidate i.e. the path with the CS containing the content.
7. Verify the selected handling application is initialized e.g. VLC, but content streaming is not possible.
8. Repeat steps 1-5 and alter type to strict.
9. Access CME terminal and confirm that there is no candidate.
10. Repeat steps 1-5 with aspiration set to 0.1 and the content should be made available to one more CS with status M.
11. Access CME terminal and confirm that there are two candidates and the selected server is the one with least overall load i.e. highest rank.

3.4.5.6 Test case 5: BW management

The consumption of content between ISPs CME depends on the availability of resources on every path. The following tests validated the selection of alternative CoS and paths/servers based on the availability of bandwidth between ISPs (domains). The tests were performed between all domains (i.e. involving all CMEs) by performing inter-domain consumptions from single and multiple CCs on v3 testbed. E.g.:

Inter-domain consumption CC 202.202.202.2 – CS 111.111.111.3

1. Repeat steps 1,2,4-6 described in previous section.
2. Open MySQL Query browser in all CMEs and record the following values from the following tables:
    a. Table *cafes_key*: COS, DELIVERY KEY, RESERVED CAPACITY, TOTAL CAPACITY
    b. Table *path_key*: BW_AGGREGATE, BW USED, CAFE IP, COS, DELIVERY KEY.
    c. Table *paths_cafes*: COS, PEER_CME_IP, RESERVED CAPACITY, SINK_CAFE_IPM SOURCE_CAFE_IP, TOTAL CAPACITY.
3. Verify that the CMEs have produced the correct key and that the provisioning process returned the correct series of keys at CME 10.1.0.6.
4. Verify that the content is streaming.
5. Repeat steps 1-4 for performing 2nd consumption.
6. Verify that the same key is produced and BW_USED has incremented by the content bit rate as registered.
7. Repeat all steps until there is no BW availability on the aggregate path.
8. Verify that no more consumption is allowed on that path and an alternative path was selected.
9. Repeat all steps until on the alternative path the BW aggregate is exceeded as well.
10. Verify that no more consumption is allowed on that path and the second source (BTBE) and equivalent path is selected.

Inter-domain consumption CC 202.202.202.2 and CC 202.202.202.22 with CS 111.111.111.3

1. Repeat previous steps for performing consumption simultaneous consumption of same content with CC 202.202.202.2 and CC 202.202.202.22.
2. Open MySQL Query browser in all CMEs and record the following values from the following tables:
   a. Table *cafes_key*: COS, DELIVERY KEY, RESERVED CAPACITY, TOTAL CAPACITY
   b. Table *path_key*: BW_AGGREGATE, BW USED, CAFE IP, COS, DELIVERY KEY.
3. Verify that the CMEs have produced the correct key and that the provisioning process returned the correct series of keys at CME 10.1.0.6.
4. Verify that the content is streaming.
5. Repeat steps 1-4 for performing multiple simultaneous consumptions.
6. Verify that in *path_key*table we have all the streams from the two CC and that RESERVED CAPACITY and BW_USED are increased appropriately.
7. Repeat all steps until there is no BW and Capacity availability.
8. Verify that no more consumption is allowed by the last CC on that path and an alternative server/path or CoS has been provided.

3.4.5.7 Results

All the above tests were performed successfully selecting the appropriate path and server parameters in all parameters examined. The results of the tests described in this section can be found in the Annex 11.2 and 12.2

3.4.6 Path configuration

3.4.6.1 Objective

Ensure that the client-side CME is capable to translate paths into keys and the server-side CME can configure the edge CAFE with the selected path i.e. forward the keys to the CAFE agent via CME-CAFE interface. The results of the test procedure described in this section can be found in the Annex 12.1.

3.4.6.2 Test case 1: CME path configuration

The following tests were repeated between several access networks using v3 testbed:

Consumption CC 101.101.101.2 – CS 111.111.111.3

1. Register content that is available in CC 111.111.111.13 (as1_rtsp_video).
2. Assign to CC 202.202.202.2 PR CoS
3. Initialize tool tcpdump on the CAFEs along the path and the alterative path on the appropriate interfaces i.e. 10.1.0.12, 11.11.11.11, 13.13.13.11 and 202.202.202.11.

   $tcpdump -i {eth0 or eth1 or eth2} -e
4. Access CME and register CC 101.101.101.2 (if not registered) with CoS PR.
5. Open Mozilla Firefox at CC 101.101.101.2 and request the content using the address bar.
6. COMET://COMET1/as1_rtsp_video
7. Verify that the content is streaming.
8. Verify that the CME 10.1.0.6 terminal has produced the correct path (i.e. key) by applying the parameters that were tested on previous releases (IPTD, IPLR etc).

9. Verify the packets travel along the path specified by CME key i.e. forwarded by border CAFEs and not the default routed path.

Consumption CC 101.101.101.2 – CS 111.111.111.3 while server-side edge CAFE is disconnected (negative scenario)

1. Repeat above steps 1-4.
2. Turn off edge CAFE agent at server-side.
3. Open Mozilla Firefox at CC 101.101.101.2 and request the content using the address bar.
4. Verify that the CME 10.1.0.6 terminal has produced an error message and consumption not satisfied.
5. Verify that no streaming was achieved and no packets travel along the path.

3.4.6.3 Results

Tests completed with expected outcomes. Also the tests involved validation of actual path forwarding that further confirmed overall successful path configuration performed by CME due to the fact that the packets travelled the path specified.

3.4.7 Content delivery

3.4.7.1 Objective

The objective of tests related to content delivery is validating the implementation of Content Aware Forwarding Entities (CAFE). We followed the bottom-up approach. We began with testing single CAFE components, i.e., *cafe_agent, cafe_forward, cafe_intercept, cf_tool, ci_tool* (see [5] for details of the components) or groups of components when simple testing of single components was not feasible. After that, we tested CAFE entity in standalone scenario; finally we tested a cooperation of several CAFEs. Apart from testing CAFE components, we simultaneously validated all its interfaces, i.e., CAFE-CAFE, CAFE-IP, CME-CAFE (see [5] for details of the interfaces). Moreover, we tested if CAFE does not disturb forwarding of standard IPv4/IPv6 packets.

The validation of CAFE consists of seven test cases, which cover all CAFE's functionalities. Each test case consists of specific tests related to one sub-functionality.

3.4.7.2 Test case 1: cf_tool

The objective of test case *1* is to validate *cf_tool* element, which is mainly responsible for managing the CAFE forwarding table. The Device Under Test (DUT) (in this case the CAFE machine) is verified manually by tester, as depicted in Figure 8.

Note that this test case does not validate if forwarding rules are properly interpreted by *cf_forward* module, which is tested in Use case #4 (see subsection 3.4.7.5).

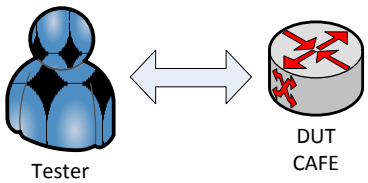

Figure 8: Test cases: *1* and *2*

3.4.7.2.1 Tests 1.1-1.3: Handling of CAFE forwarding table

The objective of the following tests 1.1-1.3 is to check if *cf_tool* properly adds and deletes entries into the CAFE forwarding table. The supported entries to be tested are standard Ethernet, VLAN Ethernet and GRE tunnels.

The testing procedure is as follows:

1. Add new entry forwarding (entryX) to CAFE forwarding table by command "*cf_tool add_ethernet a7 eth1 aa:11:b0:00:00:01*".
2. Check if forwarding table has been updated with entryX by command "*cf_tool get*".
3. Delete entryX from forwarding table by command "*cf_tool remove a7" command*".
4. Check if entryX has been deleted from forwarding table by command "*cf_tool get*".

This procedure has been performed for: standard Ethernet (Test 1.1), VLAN Ethernet (Test 1.2) and GRE (test 1.3) interfaces, see Table 11. In each test also negative scenarios were verified (e.g. addition of mal-formatted entries). In all cases properly formatted forwarding entries were successfully added and deleted. Moreover, *cf_tool* module returned an error message when it was executed with mal-formatted entries, as expected. Based on performed tests, we conclude that *cf_tool* properly handles CAFE forwarding table.

Table 11 Tests of *cf_tool*

Test id.	Forwarding technology	Expected results	Obtained results
1.1	Ethernet	Standard Ethernet entries are properly added to CAFE forwarding table. Standard Ethernet entries are properly deleted from CAFE forwarding table.	Standard Ethernet entries are properly added to CAFE forwarding table. Standard Ethernet entries are properly deleted from CAFE forwarding table.
1.2	VLAN Ethernet	VLAN Ethernet entries are properly added to CAFE forwarding table. VLAN Ethernet entries are properly deleted from CAFE forwarding table.	VLAN Ethernet entries are properly added to CAFE forwarding table. VLAN Ethernet entries are properly deleted from CAFE forwarding table.
1.3	GRE	GRE entries are properly added to CAFE forwarding table. GRE entries are properly deleted from CAFE forwarding table.	GRE entries are properly added to CAFE forwarding table. GRE entries are properly deleted from CAFE forwarding table.

3.4.7.3 Test case 2: ci_tool

The objective of test case *2* is to validate *ci_tool* element, which is mainly responsible for handling of the so-called interception rules. The interception rules define which packets should be intercepted and encapsulated with COMET header. The Device Under Test - DUT (CAFE machine) is verified manually by tester, as depicted in Figure 8.

Note that this test case does not validate if interception rules set with *ci_tool* are properly interpreted by *ci_forward* module. This is tested in subsection 3.4.7.6.

3.4.7.3.1 Test 2.1: Manual addition and deletion of interception rules

The objective of the test 2.1 is to validate manual addition and deletion of interception rules.

The testing procedure is as follows:

1. Add a new interception rule (ruleX) using ci_tool setting timeout value for this rule, e.g. by command "*ci_tool add 123 60 1.1.1.2 2.2.2.1 17 0 80 1234567890abcdef*".
2. Check if interception rule was properly added, i.e. check if no errors have been retuned by ci_tool.
3. Delete ruleX by ci_tool before the timeout for this rule expires, e.g., by command "*cf_tool remove 123*".
4. Check if interception rule was properly deleted, i.e., check if no errors have been retuned by ci_tool.

This test has been performed for positive scenario (properly formatted parameters of *ci_tool command*) as well as for negative scenario (mal-formatted parameters of *ci_tool command*). In the positive scenario interception rules were properly added and deleted, as expected, whereas in the negative scenario *ci_tool* element returned an error, as expected. Based on performed tests, we conclude that *ci_tool* properly handles manual addition and deletion of interception rules.

3.4.7.3.2 Test 2.2: Manual addition and automatic deletion of interception rules

The objective of the test 2.1 is to validate the manual addition and deletion of interception rules. The interception rule should be automatically deleted after expiration of timeout (timeout is configured during the addition of the interception rules).

The testing procedure is as follows:

1. Add a new interception rule (ruleX) using ci_tool, setting timeout value for this rule, e.g. by command "*ci_tool add 123 60 1.1.1.2 2.2.2.1 17 0 80 1234567890abcdef*".
2. Check if interception rule was properly added, i.e. check if no errors have been retuned by ci_tool.
3. Wait until timeout expires.
4. Check if the interception rule was automatically deleted, e.g., by command "*ci_tool collect*".

Interception rules were properly added and deleted (after expiration of timeout) as expected. As a consequence of performed tests, we conclude that *ci_tool* properly handles manual addition and automatic deletion of interception rules.

3.4.7.4 Test case 3: cafe_agent

The objective of the *test case 3* is to validate cafe_agent element and CAFE-CME interface. The cafe_agent element enables CME to remotely configure streams (configure interception rules) and to collect statistics of expired streams. In order to validate cafe_agent's implementation, a special script has been written and used at the Automatic Test Equipment - ATE, see Figure 9. This script performs the following actions:

1. connects to CAFE,

2. collects expired streams,
3. configures new streams (ten sample streams).

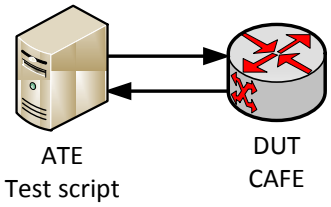

Figure 9: Test case *3*

3.4.7.4.1 Test 3.1: Configuration of streams and collection of expired streams

The objective of this test is to check if cafe_agent can remotely configure streams and collect statistics.

The testing procedure has the following steps:

1. running test script,
2. analysing results.

The expected results are

- expired streams are shown,
- sample streams are configured

The results obtained were the expected ones, so we conclude that cafe_agent is implemented correctly.

3.4.7.5 Test case 4: cf_forward

The objective of this test case is to validate *cf_forward* module and CAFE-CAFE interface. The *cf_forward* module is responsible for forwarding COMET packets based on the keys included in the COMET header. The Spirent TestCenter [12] is used as Automatic Test Equipment – ATE, as depicted in Figure 10. The ATE is connected to DUT by three Ethernet links, first link (*link_a*) is used to transfer packets from ATE to DUT, other two links (*link_b* and *link_c*) are used to transfer packets back from DUT to ATE.

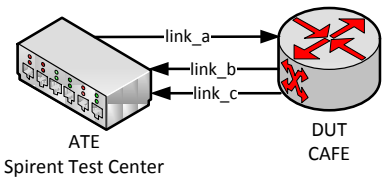

Figure 10: Test cases: *4* and *5*

3.4.7.5.1 Test 4.1: Forwarding of COMET packets based on forwarding key via standard Ethernet interfaces

The objective of this test is to validate if CAFE properly forwards COMET frames via standard Ethernet interfaces. The frames are forwarded according to the COMET forwarding table. The ATE generates two streams of COMET packets, with different COMET headers, encapsulated into standard Ethernet frames. These two streams are sent by link_a. The COMET headers (see [3] for details) of the packets from the first stream are set with the following values of fields: *Length* equal to "0x02", *Index* equal to "0x00" and *List of forwarding keys* equal to "0xA1A2". The COMET headers in packets from the second stream are set with following values of fields: *Length* equal to "0x03", *Index* equal to "0x01" and *List of forwarding keys* equal to "0xB1B2B3". Thus, packets from the first stream are forwarded by CAFE on the basis of the first key, and packets from the

second stream are forwarded on the basis of the middle key. The expected actions of CAFE is to send the packets of the first stream by link_b and the packets of the second stream by link_c.

The testing procedure is as follows:

1. Configure CAFE with three standard Ethernet interfaces: eth0 connected to link_a, eth1 connected to link_b and eth3 connected to link_c.
2. Configure cf_forward module with forwarding keys, egress interfaces and destination Ethernet addresses, frames with key "A1" should be forwarded via interface eth1 and frames with key "A2" should be forwarded via interface eth2.
3. Generate two preconfigured streams of COMET packets from ATE to DUT via link_a.
4. Capture frames on ATE's interfaces connected to link_b and link_c.
5. Analyse captured frames.

Expected results are as follows:

- Frames are forwarded by CAFE via proper egress interfaces, i.e. the frames from first stream are forwarded via *link_b* and the frames from the second stream are forwarded via *link_c*.
- Captured frames have modified: COMET header (*Index* filed is incremented), source MAC address (source MAC address is set with egress CAFE interface's MAC address) and destination MAC address (destination MAC address is set as defined in step 2).

The results obtained were as expected, so we conclude that CAFE properly forwards COMET packets via standard Ethernet interfaces. In this test, the packets were forwarded on the basis of COMET header in the IP version agnostic way (nevertheless we performed this test for IPv4and IPv6 datagrams encapsulated with COMET header).

Moreover we repeated this test with background IP traffic (both IPv4 and IPv6) traversing through CAFE to investigate if handling COMET traffic disturbs handling of IP traffic (and vice versa). As a result of performed tests we conclude that handling of COMET traffic does not disturb handling of IP traffic (and vice versa).

3.4.7.5.2 Test 4.2: Forwarding of COMET packets based on forwarding key via VLAN Ethernet interfaces

The objective of this test is to validate if CAFE properly forwards COMET frames via VLAN Ethernet interfaces. The packet streams generated by ATE are almost the same as described in 3.4.7.5.1, the only difference is that in this case COMET packets are encapsulated into VLAN Ethernet frames (instead of standard Ethernet frames). All the packets generated by ATE have VLAN tag set to "0x01".

The testing procedure is as follows:

1. Configure CAFE with three standard Ethernet interfaces: eth0 connected to link_a, eth1 connected to link_b and eth2 connected to link_c.
2. Configure CAFE with three virtual VLAN Ethernet interfaces: eth0.01 (VLAN "01") connected to link_a, eth1.10 (VLAN "10") connected to link_b and eth2.20 (VLAN "20") connected to link_c.
3. Configure cf_forward module with forwarding keys, egress interfaces and destination Ethernet addresses, frames with key "A1" should be forwarded via interface eth1.10 and frames with key "A2" should be forwarded via interface eth2.20.
4. Generate two preconfigured streams of COMET packets from ATE to DUT via link_a.
5. Capture frames on ATE's interfaces connected to link_b and link_c
6. Analyse captured frames.

Expected results are as follows:

- Frames are forwarded by CAFE via proper egress interfaces, i.e. frames form first stream are forwarded via interface *eth1* and frames from the second stream are forwarded via interface *eth2*.
- Captured frames have modified 802.1q header with proper VLAN, i.e .frames forwarded via *link_b* are tagged with vlan "10" and frames forwarded via *link_b* are tagged with vlan "20".
- Captured frames have modified: COMET header (*Index* filed is incremented), source MAC address (source MAC address is set with egress CAFE interface's MAC address) and destination MAC address (destination MAC address is set as defined in step 1).

The results were as expected, so we conclude that CAFE properly forwards COMET packets via VLAN Ethernet interfaces. In this test, the packets were forwarded on the basis of COMET header in the IP version agnostic way.

3.4.7.5.3 Test 4.3: Forwarding of COMET packets based on forwarding key via GRE tunnels

The objective of this test is to validate if CAFE properly forwards COMET frames via GRE tunnel. The packet streams generated by ATE are almost the same as described in 3.4.7.5.1, the only difference is that in this case COMET packets are encapsulated with GRE header (instead of standard Ethernet frames).

The testing procedure is as follows:

1. Configure CAFE with three standard Ethernet interfaces: eth0 connected to link_a, eth1 connected to link_b and eth2 connected to link_c.
2. Configure CAFE with three GRE interfaces: gre01 connected to link_a, gre02 connected to link_b and gre03 connected to link_c.
3. Configure cf_forward module with forwarding keys, egress interfaces and destination Ethernet addresses, frames with key "A1" should be forwarded via interface gre02 and frames with key "A2" should be forwarded via interface gre03.
4. Generate two preconfigured streams of COMET packets from ATE to DUT via link_a.
5. Capture frames on ATE's interfaces connected to link_b and link_c
6. Analyse captured frames.

Expected results are as follows:

- Frames are forwarded by CAFE via proper egress interfaces, i.e. frames from the first stream are forwarded via *link_b* and frames from the second stream are forwarded via *link_c*.
- Captured frames have modified GRE (according to configuration of GRE interfaces).
- Captured frames have modified: COMET header (*Index* filed is incremented), source MAC address (source MAC address is set with egress CAFE interface's MAC address) and destination MAC address (destination MAC address is set as defined in step 1).

The obtained results were as expected, so we conclude that CAFE properly forwards COMET packets via GRE Ethernet interfaces. In this test packets were forwarded on the basis of COMET header in the IP version agnostic way.

3.4.7.5.4 Tests 4.4 – 4.5: Decapsulation of COMET packets

The objective of this tests is to validate if CAFE properly decapsulates COMET packets. The test topology consists of ATE (Spirent TestCenter) and DUT (CAFE) as depicted in Figure 10 (in this test only *link_a* and *link_b* are used). The ATE generates one stream of packets and sends them via *link_a*. Packets have the following encapsulation: IP header, COMET header, Ethernet header. For test 4.4 ATE generates IPv4 datagrams and for test 4.5 ATE generates IPv6 datagrams. The COMET header in all packets is set with the following values of fields: *Length* equal to "0x02", *Index* equal to "0x02" and *List of forwarding keys* equal to "0xA1A2". Since *Index* field is equal to

Length field then the COMET header should be removed from frames by CAFE. As a consequence packets should be forwarded on the basis of IP header.

The testing procedure for both tests is as follows:

1. Configure CAFE with two standard Ethernet interfaces: eth0 connected to link_a and eth1 connected to link_b.
2. Configure default routing: IPv4 routing in test 4.4 and IPv6 routing in test 4.5 (all IPv4/IPv6 frames should be forwarded via DUT's interface eth1).
3. Generate preconfigured stream of COMET packets from ATE to DUT via link_a.
4. Capture frames on ATE's interface connected to link_b.
5. Analyse captured frames.

Expected results are as follows:

- Frames are forwarded by CAFE via interface eth1 (on the basis of IP routing).
- COMET header is removed from frames.
- Ethernet header is modified (source MAC address is set with egress CAFE interface's MAC address) and destination MAC address (destination MAC address is set as defined in step 1).

The results obtained were as expected, so we conclude that CAFE properly decapsulates COMET packets.

3.4.7.6 Test case 5: cf_intercept

The objective of this test case is to validate *cf_intercept* module and CAFE-IP router/terminal interface. The *cf_intercept* module is responsible for encapsulation of IP packets with COMET header based on preconfigured interception rule. The Spirent TestCenter is used as ATE, as depicted in Figure 10. In this test case *link_a, link_b* and *link_c* are exploited: *link_a* is used to transfer packets from ATE to DUT, *link_b* and *link_c* are used to transfer packets back from DUT to ATE.

3.4.7.6.1 Tests 5.1 - 5.20: Interception of IPv4/IPv6 datagrams

The objective of this test is to validate if CAFE intercepts IP datagrams according to the configured interception filters and encapsulates them with COMET header.

For each test, the ATE generates one stream of IP datagrams with the following encapsulation: Ethernet header, IPv4/IPv6 header, UDP header. The CAFE is configured with default routing, so that all IP packets are forwarded via interface eth2. The *cf_intercept* module is configured for each test according to Table 12, where:

- The "Proper" value of given field means that the value of filter matches with the value of corresponding field in the frames generated by ATE.
- The "Wrong" value of given field means that the value of filter does not match with the value of corresponding field in the frames generated by ATE.
- The "Any" value of field means that this filter is disabled.

Moreover for all tests:

- *cf_intercept* is configured to encapsulate IP datagrams into COMET header with *List of forwarding keys* field equal to "0xA1A2",
- *cf_forward* is set to forward COMET frames with key "A1" should via interface eth1.

As a consequence:

- all IP packets incoming to CAFE and matching interception filter should be encapsulated with COMET header and forwarded via eth1 interface,
- all IP packets incoming to CAFE not matching any interception filter should be forwarded via eth2 interface.

The testing procedure is as follows:

Pre-configuration (done once for all tests):

1. Configure CAFE with three standard Ethernet interfaces: eth0 connected to link_a, eth1 connected to link_b and eth2 connected to link_c.
2. Configure default routing (all IPv4/IPv6 frames should be forwarded via DUT's interface eth2).
3. Configure cf_forward module with forwarding keys, egress interface and destination Ethernet address.

For each test:

1. Configure cf_intercept module according to Table 12.
2. Generate packet stream from ATE to DUT via link_a.
3. Capture frames on ATE's interfaces connected to link_b and link_c.
4. Analyse captured frames.

Table 12 Tests of *cf_intercept*

Test id.	cf_intercept configuration						Results	
	IP	Src IP	Dst IP	Prot	Src port	Dst port	Expected result	Obtained result
5.1	IPv4	Proper	Proper	Any	Any	Any	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1
5.2	IPv4	Proper	Wrong	Any	Any	Any	Frames are forwarded via eth2	Frames are forwarded via eth2
5.3	IPv4	Wrong	Proper	Any	Any	Any	Frames are forwarded via eth2	Frames are forwarded via eth2
5.4	IPv4	Proper	Proper	Proper	Any	Any	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1
5.5	IPv4	Proper	Proper	Wrong	Any	Any	Frames are forwarded via eth2	Frames are forwarded via eth2
5.6	IPv4	Proper	Proper	Any	Proper	Any	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1
5.7	IPv4	Proper	Proper	Any	Wrong	Any	Frames are forwarded via eth2	Frames are forwarded via eth2
5.8	IPv4	Proper	Proper	Any	Any	Proper	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1
5.9	IPv4	Proper	Proper	Any	Any	Wrong	Frames are forwarded via eth2	Frames are forwarded via eth2
5.10	IPv4	Proper	Proper	Proper	Proper	Proper	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1

5.11	IPv6	Proper	Proper	Any	Any	Any	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1
5.12	IPv6	Proper	Wrong	Any	Any	Any	Frames are forwarded via eth2	Frames are forwarded via eth2
5.13	IPv6	Wrong	Proper	Any	Any	Any	Frames are forwarded via eth2	Frames are forwarded via eth2
5.14	IPv6	Proper	Proper	Proper	Any	Any	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1
5.15	IPv6	Proper	Proper	Wrong	Any	Any	Frames are forwarded via eth2	Frames are forwarded via eth2
5.16	IPv6	Proper	Proper	Any	Proper	Any	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1
5.17	IPv6	Proper	Proper	Any	Wrong	Any	Frames are forwarded via eth2	Frames are forwarded via eth2
5.18	IPv6	Proper	Proper	Any	Any	Proper	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1
5.19	IPv6	Proper	Proper	Any	Any	Wrong	Frames are forwarded via eth2	Frames are forwarded via eth2
5.20	IPv6	Proper	Proper	Proper	Proper	Proper	Frames are encapsulated with COMET header end forwarded via eth1	Frames are encapsulated with COMET header end forwarded via eth1

The results obtained were the expected ones, so we conclude that CAFE properly intercepts and IPv4/IPv6 datagrams and properly encapsulates them with COMET header.

In conclusion, the following CAFE elements have been used/tested simultaneously: *cf_forward, cf_intercept, cf_tool, ci_tool.* Because of positive results we conclude that standalone CAFE is working properly.

3.4.7.7 Test case 6: Cooperation of CAFEs

The objective of this test case is to review cooperation of several CAFEs and to validate CAFE-IP interface. The test topology consists of ATE (Spirent TestCenter) and three DUTs (CAFEs) as depicted in Figure 11. The $CAFE_A$ intercepts incoming IP datagrams, encapsulates them with COMET header and forwards them to $CAFE_B$ (accordingly to COMET forwarding table). The $CAFE_B$ forwards packets to $CAFE_C$ (accordingly to COMET forwarding table), finally $CAFE_C$ decapsulates IP datagrams and forwards them back to ATE (accordingly to IP forwarding table).

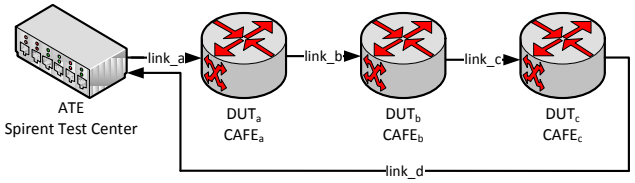

Figure 11: The cascade of CAFEs

3.4.7.7.1 Tests 6.1-6.3: Interception, forwarding and decapsulation of IPv4/IPv6 datagrams

The ATE generates:

- one stream of IPv4 datagrams in test 6.1,
- one stream of IPv6 datagrams in test 6.2,
- one stream of IPv4 datagrams and one stream of IPv4 datagrams in test 6.3.

The objective of tests:

- validation of cooperation of CAFEs handling IPv4 traffic – test 6.1,
- validation of cooperation of CAFEs handling IPv6 traffic – test 6.2,
- validation of cooperation of CAFEs handling simultaneously IPv4 and IPv6 traffic – test 6.3.

The testing procedure is as follows:

Pre-configuration (done once for all tests): We must:

1. Configure CAFEa with: two standard Ethernet interfaces: eth0 connected to link_a, eth1 connected to link_b.
2. Configure CAFEb with: two standard Ethernet interfaces: eth0 connected to link_b, eth1 connected to link_c
3. Configure CAFEc with: two standard Ethernet interfaces: eth0 connected to link_c, eth1 connected to link_d
4. Configure cf_forward module in CAFEa with forwarding keys, egress interface and destination Ethernet address (frames with key a1 should be forwarded via interface eht1).
5. Configure cf_forward module in CAFEb with forwarding keys, egress interface and destination Ethernet address (frames with key a2 should be forwarded via interface eht1).
6. Configure CAFEc with default routing (all IPv4/IPv6 frames should be forwarded via interface eth1).

For each test, we must:

1. Configure cf_intercept filter in CAFEa to match stream(s) generated by ATE
2. Generate packet stream(s) from ATE to DUT via link_a.
3. Capture frames on ATE's interface connected to link_d
4. Analyse captured frames.

Expected results are as follows:

- Frames are properly encapsulated with COMET header, forwarded on the basis of COMET header, decapsulated (from COMET header) and forwarded on the basis of IPv4/IPv6 header.
- COMET header is removed from the frames received by ATE.
- Ethernet header is modified in the frames received by ATE (source MAC address is set with egress CAFE interface's MAC address) and destination MAC address (destination MAC address is set as defined in step 3 of pre-configuration.

Obtained results were the same as expected once, so we conclude that CAFEs cooperate properly and CAFE-IP is implemented correctly.

3.4.7.8 Test case 7: Cooperation of CAFEs with COMET system

The objective of this test case is to check: (1) if CAFEs properly cooperates with other COMET entities, i.e.: CC, CS and CME, and (2) if CAFEs properly intercept and forward content. This test case has been performed in the integration testbed presented in Figure 6. We performed two tests. First test corresponds to intra-domain content forwarding and the second test corresponds to inter domain content forwarding.

3.4.7.8.1 Test 7.1: Intra-domain content forwarding

The following procedure was performed between all access networks present in the same domain using v3 testbed (see section 3.2.2.3).

Consumption CC 101.101.101.2 – CS 111.111.111.3

1. Register content that is available in CS 111.111.111.13 (as1_rtsp_video).
2. Assign to CC 101.101.101.2 PR CoS
3. Access edge CAFE 111.111.111.12 and 101.101.101.11 and initialize tool tcpdump.

   $tcpdump -i eth1 -e
4. Access CME and register CC 101.101.101.2 with CoS PR.
5. Open Mozilla Firefox at CC 101.101.101.2 and request the content using the address bar.

   COMET://COMET1/as1_rtsp_video
6. Record from CME 10.1.0.6 terminal the generated series of key.
7. Verify that the content is streaming.
8. Verify the packets are intercepted and forwarded appropriately by examining the header.

This test confirmed that content could be forwarded between adjacent edge CAFEs (i.e. access networks) in the same domain. The results are as shown in the Annex 12.3

3.4.7.8.2 Test 7.1: Intra-domain content forwarding

The following procedure was performed between all access networks present in the different domains (i.e. between access networks located in remote domains) using v3 testbed.

Consumption CC 202.202.202.2 – CS 111.111.111.3

1. Register content that is available in CC 111.111.111.13 (as1_rtsp_video).
2. Assign to CC 202.202.202.2 PR CoS
3. Initialize tool tcpdump on the CAFEs along the path and the alterative path on the appropriate interfaces i.e. 10.1.0.12, 11.11.11.11, 13.13.13.11 and 202.202.202.11.

   $tcpdump -i {eth0 or eth1 or eth2} -e
4. Access CME and register CC 101.101.101.2 (if not registered) with CoS PR.
5. Open Mozilla Firefox at CC 101.101.101.2 and request the content using the address bar.
6. COMET://COMET1/as1_rtsp_video
7. Record from CME 10.1.0.6 terminal the generated series of key.
8. Verify that the content is streaming.
9. Verify that the packets are intercepted and that the packets now contain the COMET headers at 10.1.0.12 (o/p of CAFE edge).

10. Verify the packets travel along the path specified by CME key (i.e. forwarded by border CAFEs.

This test finalised the content forwarding tests and ensured that content could be forwarded in all domains and access networks. The results are as shown in the Annex 12.3

3.5 Overall validation of decoupled prototype

The system was tested in two phases, during partial integration prior to release 2 using v2 testbed and when fully integrated prior to the final release (release 3) using v3 testbed. The same test cases were applied on both phases when performing publication and consumption of live and pre-recorded content in intra or inter domain scenarios.

3.5.1 Intra-domain

The following tests confirmed that content was deliverable in the same domain network. The steps performed in order to request content locally (e.g. in domain 1) were as shown in the below example:

Consume content *domain_1_video_rtsp*

1. Access CME administration at IP 10.1.0.6 by accessing the administration page at http://localhost:8090/CME/main.jsf
2. Create authorized CC with CoS PR and IP 10.1.0.2
3. Initialize a terminal at CC VM and change directory to the CC executable.
4. Request a published content as follows:

   Client 10.1.0.6 9091 domain_1_video_http
5. Verify that the correct content has been delivered.

3.5.2 Inter-domain

The following test confirmed that a content that not mediated only by the local CME and equivalently not registered to the local CRE was deliverable to any CC at any remote domain. The tests involved consumption in-between all 3 domains. The steps performed in order to request content remotely were as shown in the below example:

Domain 1 CC requests content available in domain 2 CS:

1. Create content as described in section 4.1.1 for domain 1.
2. Access CC at 10.2.0.2
   a. Initialize a terminal at CC VM and change directory to the CC executable.
3. Request the content created in step 1 as follows:

   Client 10.2.0.6 9091 {content name}
4. Verify that the correct content has been delivered.

3.5.3 Live content

Live content was consumed for confirming that the system could cope with real-time requirements. The steps were performed as shown in the below example:

Domain 1 CC requests RTSP content available in domain 2 CS

1. Create content as described in section 4.1.1 for domain 2 with a suitable protocol (RTSP).
2. Access CS at 10.2.0.3.
3. Initialise VLC and assign a video file for streaming.
4. Request the content via CC as described in previous tests.
5. If requested, provide associated application for the content's application protocol.
6. Verify the selected handling application is initialized i.e. VLC, and the correct content is streaming.

3.5.4 Results of overall validation

The COMET system as a whole produced no failed attempts of content consumption by the client for any type of content and network environment. Thus, the overall validation finalised and ensured the proper functionality of the system. The test results and further detail for the overall validation can be found in the Annex 11.4 and 12.5.

3.6 Summary

Section 3 describes in depth the work performed in order to integrate and validate the decoupled COMET prototype.

As described in section 3.2 the COMET system was build using a variety of programming languages and tools, and entity by entity was integrated in larger and larger topologies in coherence to the phase of development. Thus, using an iterative development approach the system was finalised and integrated completely in three major releases described in section 3.3.

The COMET decupled prototype and particular entities and processes were validated prior to each release using the equivalent testbeds described in section 3.2.2. As described in section 3.4 several positive and negative scenario tests were performed that ensured the proper functionality of individual and/or group of entities together with their communicating interfaces. The tests performed for each main COMET system functions are allocated in sub-section 3.4.1-3.4.7. Finally following the functional testing the system was overall tested as described in section 3.5. The results are satisfactory.

4 Integration and validation of coupled proof-of-concept prototype

4.1 Introduction

In this section we discuss in detail the integration and validation process of the COMET coupled approach. Our integration and validation process involved a step-wise progress that started with the validation of single entities from *"correct handling of registration/publication messages"* to *"correct handling of received content"*. The detailed process, as well as the steps and the output of our prototype are presented in detail in Table 13, in Section 4.2.

In Section 4.3, we present the testing process of our integrated prototype. In particular, we first present in detail the steps during the implementation process (i.e., the implementation sequence of different functions for each entity). This process is also given in a waterfall diagram Figure 13. Then, in the same section, we also present the testing approach of the various content lifetime stages which are carried out within the context of different test topologies. These stages are split into *Content Publication, Content Resolution, Content State Installation* and the *Content Delivery*, and are described in Sections 4.3.2.1 to 4.3.2.3, respectively.

Section 4.4 gives the overall validation of the coupled approach proof-of-concept prototype. Finally, we summarise in Section 4.5.

4.2 Validation of entities

Validation of the various COMET entities in the coupled approach was carried out and documented in Table 13. These tests cover the CRME, CAFE, CC and CP, and were based on the simple topology shown in Figure 12.

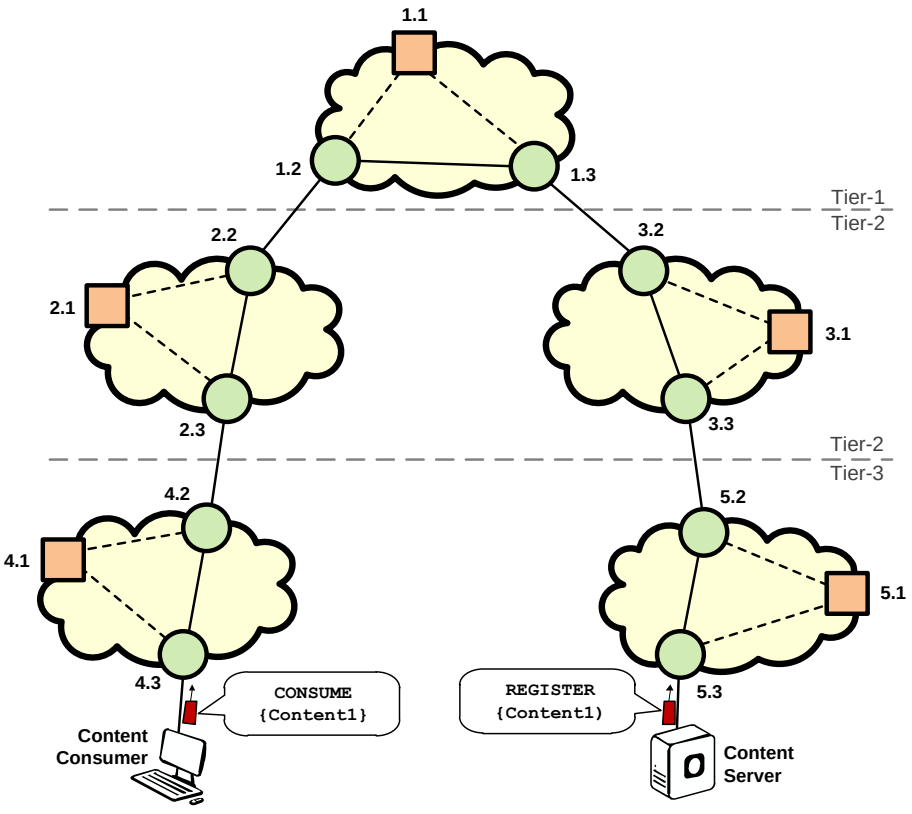

Figure 12: Test topology 1.

Table 13: Validation of coupled approach COMET entities

Test Category	Test	TimeStamp/ Owner	Test Description	Steps	Output / Result	Result
1. Content resolution and mediation entity (CRME)	1	10-01-2012/georgekamel	Correct handling of a content registration/publication message	Create a new entry in the content table Forward the register message as a publish message to CRMEs in provider domains	CRME: Register message for Content 'Content1' received from address 5.4. CRME: Content ID 'Content1' added to ContentTable. CRME: Publish message sent to CRME [3.1] in provider domain.	PASS
	2	10-01-2012/georgekamel	Ability to handle multiple content registrations/publications	Same as above, but repeated over a number of content publications	Above output was repeated for each content being registered.	PASS
	3	10-01-2012/georgekamel	Correct forwarding of content publication message in the presence of a peering link	Create a new entry in the content table Forward the register message as a publish message to CRMEs in provider domains, and not along peering link	CRME: Register message for Content 'Content1' received from address 5.1. CRME: Content ID 'Content1' added to ContentTable. CRME: Publish packet sent to CRME [1.1] in provider domain.	PASS
	4	10-01-2012/georgekamel	Correct forwarding of a content consumption message in the	Check if entry for the requested content is contained in the content table	CRME: ContentID 'Content1' not found in ContentTable. Consume message will be forwarded to a provider CRME [1.1]	PASS

Test Category	Test	TimeStamp/ Owner	Test Description	Steps	Output / Result	Result
			presence of a peering link, in the case where content ID is not contained in table	Forward consume packet to CRME in provider domain and not in peering domain		
	5	10-01-2012/georgekamel	Correct forwarding of a content consumption message when an entry for the requested content is present in the CRME's content table	Check that an entry for the requested content is contained in own content table	`CRME: ContentTable contains entry for ContentID 'Content1'.`	**PASS**
				Forward consume packet to CRME or content publisher according to entry in content table	`CRME: Consume message will be forwarded to 5.1:3000.`	
2. Content-Aware Forwarding Entity (CAFE)	**6**	10-01-2012/georgekamel	Correct handling of content state configuration message	Receipt of configure message	States are installed correctly in the CAFE and the entire content state table of that CAFE is displayed as follows:	**PASS**

				Installation of state in content state table	`CAFE ContentStateTable at 2.2:`
					`Content ID | Next Hop Address | Next Hop Port`
					`MyContent1 2.3 2000`	

	7	10-01-2012/georgekamel	Correct forwarding of received content packets	Look up next hop CAFE or content client from content state table	`CAFE: Content 'Content1' received and will be sent to 2.3:2000`	**PASS**
				Transmit content to next hop		
3. Content Publisher (CP)	**8**	10-01-2012/georgekamel	Forwarding of register message to local CRME	Construct register message	`CP: Registering content Content1 with local CRME [5.1]`	**PASS**
				Transmit message to local CRME		

Test Category	Test	TimeStamp/ Owner	Test Description	Steps	Output / Result	Result
	9	10-01-2012/georgekamel	Correct handling of content consumption message and forwarding of content to correct next hop	Receipt of consume message	CP: Consume message for Content 'Content1' received at ContentPublisher CP: Content 'Content1' will be sent to 5.3:2000	PASS
				Determine next hop from consume message		
				Construct datagram containing content		
				Transmit content to next-hop CAFE towards content client		
4. Content Client (CC)	10	10-01-2012/georgekamel	Forwarding of consume message to correct CRME	Construct consume message	CC: Sending Consume message for Content 'Content1' to local CRME [4.1]	PASS
				Transmit message to local CRME		
	11	10-01-2012/georgekamel	Correct handling of received content	Receipt of content message	CC: Content 'Content1' received and saved to: ./Content/4.4 1000/Content1	PASS
				Store content to disk		

4.3 Integrated test environment

4.3.1 The integrated implementation approach

From the outset, the coupled approach implementation within the VLNSP has adopted a waterfall approach, as shown in Figure 13, wherein the various entities were implemented sequentially. For each entity, the control-plane functionality (content registration, publication and resolution) was first implemented. The implementation was then extended with data-plane functionality (content delivery). This approach has ensured that the system works as a whole without the need for a specific integration phase.

4.3.2 Test topologies and scenarios

The testing of the system was carried out on each of the basic individual features, namely content registration/publication, content resolution, and content delivery. The following subsections detail each of the tests carried out, including the validation criteria of each test.

4.3.2.1 Basic content publication operation

To test the content publication operation, an inter-domain topology shown in Figure 12 and Figure 14 are setup. The test simply involves a content server attached to 5.3 registering different content (i.e., issuing `Register` messages for different content).

In summary our tests validate the following:

- The correct forwarding of `Register` messages from CP to its local CRME (via intermediate CAFE(s))
    - The `Register` message should traverse from CP to node 5.1 via node 5.3 for test topology 1 and node 7.1 via node 7.3 for test topology 2.
- The correct handling of `Register` messages at the local CRME (i.e., node 5.3 in test topology 1 and node 7.3 in test topology 2)
    - The local CRME should be able to create a new entry in its `contentTable` for each new content being registered.
- The correct creation of the `Publish` messages from local CRME
    - The content ID must be the same as the corresponding `Register` message.
- The correct forwarding of the `Publish` messages
    - Test topology 1: The `Publish` messages should be forwarded following the *provider route forwarding rule* (i.e., node 5.1 → node 5.2 → node 3.3 → node 3.1 → node 3.2 →node1.3 → node 1.1)
    - Test topology 2: The `Publish` messages should be forwarded following the *provider route forwarding rule* (i.e., node 7.1→ node 7.2 → node 4.3 → node 4.1 → node 4.2 →node 1.4→ node 1.1)
- The correct handling of `Publish` messages at each CRME
    - Each CRME receiving the `Publish` messages should be able to create a new entry in its `contentTable` for each new content being published.

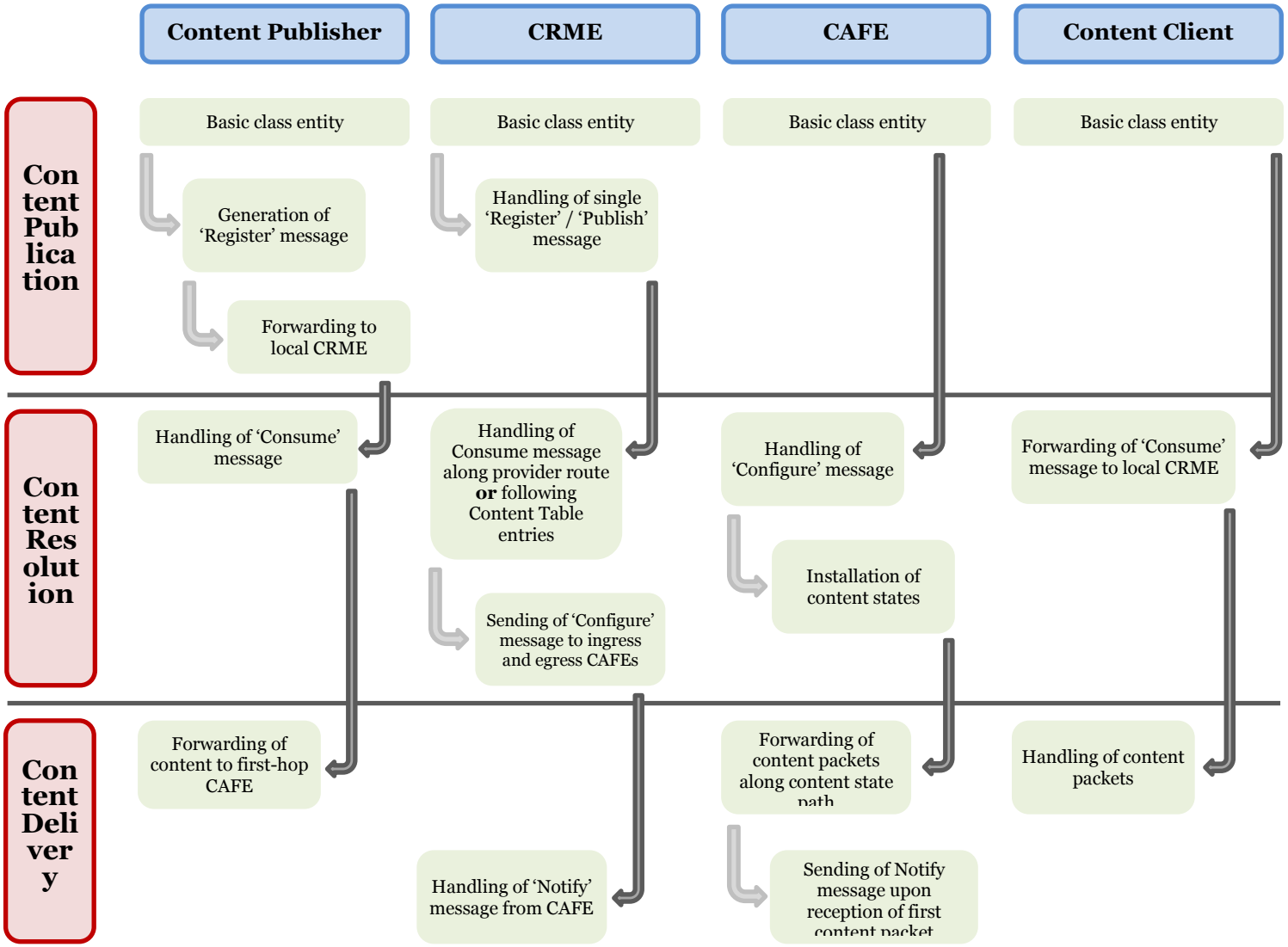

Figure 13: Waterfall implementation approach within the VLNSP

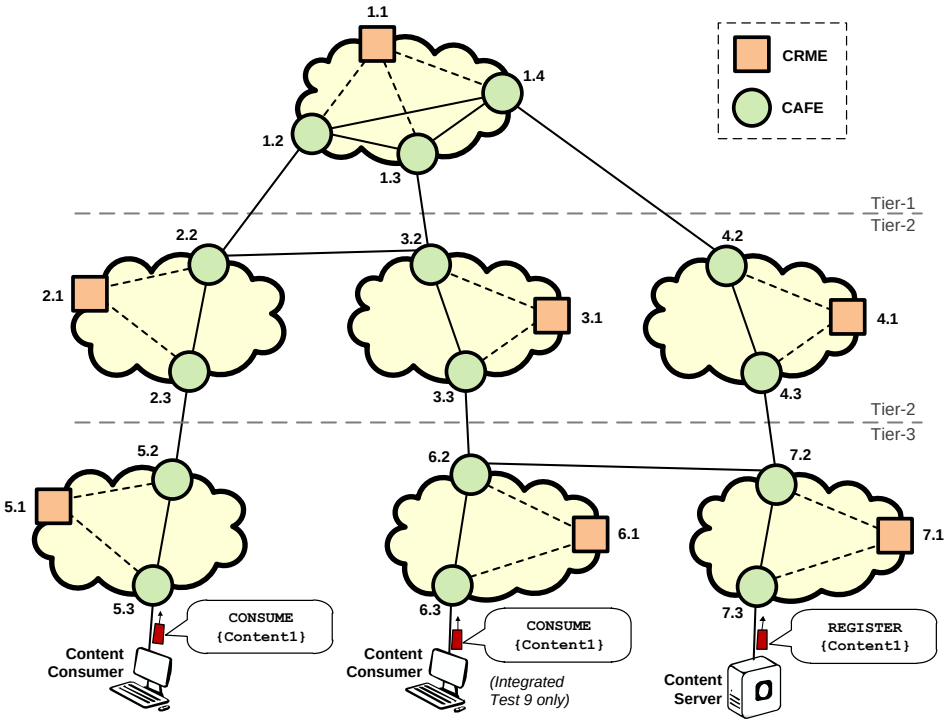

Figure 14: Test topology 2.

4.3.2.2 Basic content resolution operation

To test the content resolution operation, an inter-domain topology shown in Figure 14 is setup. It is initialised with a single content (i.e., `Content1`) being published as shown in the figure. The test involves attaching a CC at different points of the network and requesting for the content.

This enables the validation of the following:

- The correct forwarding of `Consume` messages from CC to the CP via intermediate CRME(s)
  - The `Consume` message should follow the *provider route forwarding rule* while the requested content is still not found.
  - The `Consume` message should follow the entry in the CRME's `contentTable` once the requested content is found.

4.3.2.3 Basic content state installation and delivery operation

To test the basic content delivery operation, the inter-domain topologies shown in Figure 12 and Figure 14 will be used. These test topologies involve a content consumer attached at one point of the network issuing a request from content hosted at another point within the network which happens to be already published to the COMET system. The tests aim to validate that

- Content states are installed only in the appropriate CAFEs. So, for the tier-1 domain in the case of the topology shown in Figure 14, no content states should be installed in CAFE 1.3 or in any CAFEs under it.
- The correct handling of `Consume` messages at the content client.
- The requested content passes only through the CAFEs in which a corresponding content state is present.
- Content is delivered successfully from content server to content client, and stored correctly at the content client.

Content delivery paths are altered if more optimal routes are detected.

4.4 Overall validation

This table details the validation procedures and results for basic content state installation and delivery, which was carried out using the implementation of the proof-of-concept emulator. These tests were carried out based on the two topologies shown in the previous subsection (Figure 12 and Figure 14).

We describe each test in Table 5 in terms of the expected operations to validate the correct publication, state installation and delivery of content.

Test 1: Publication of a single content item (Topology 1, Figure 12). In theory, we expect the following steps, which are indeed validated as shown in Table 5, Test 1.

- Step 1: The CP forwards the Registration message to the local CRME 5.1 at port 3000.
- Step 2: Receipt of the Registration message at CRME 5.1, update of the Content Table at 5.1 and finally, generation of the actual Publish message at CRME 5.1
- Step 3: The Publish message generated at CRME 5.1 is forwarded upwards towards all parent CRMEs, that is CRMEs 2.1 and 1.1 in our case.

Test 2: Publication of multiple content items (Topology 2, Figure 14). We expect the following steps, as indeed happens as shown in Table 5, Test 2. In essence, the previous three steps are repeated for each content item (i.e., Content1 and Content2).

- Step 1: The CP sends the Registration message for the first item (Content1) to the local CRME 7.1 at port 3000.
- Step 2: The local CRME receives the Registration message, updates its ContentTable and generates the publish message for Content1
- Step 3: The local CRME forwards the Publish message it has just generated upwards towards all parent CRMEs, in our case 4.1 and 1.1 (Topology 2, Fig. 12).
- Step 4: Step 1 above is repeated for the second content that the CP wants to publish, i.e., sending of Registration message to local CRME.
- Step 5: Step 2 above is repeated, i.e., local CRME receives the Registration message, updates its ContentTable and generates the Publish message.
- Step 6: Finally, the Publish message generated at the local CRME 7.1 for the second content (Content2) is forwarded upwards towards parent CRMEs 4.1 and 1.1

Test 3: Request for non-existent content in Topology 1.

- Step 1: The content client (CC) requests for Content2 to its local CRME 4.1 (Consume message is generated and sent.)
- Step 2: Local CRME does not have an entry for the requested content, hence, forwards the request upwards to its parent CRMEs, that is 2.1 and 1.1. CRME 1.1 returns an Error Message: Content Not Found, as this is the topmost (Tier-1) ISP.

Test 4: Request for Existent Content in Topology 2.

- Step 1: The Content Client (5.3, port 1000) forwards the Consume message for Content1 to its local CRME at 5.1, port 3000.
- Step 2: Local CRME receives the Consume request for Content1, but does not have an entry for this content in its ContentTable, hence forwards upwards to CRME 2.1. CRME 2.1 does not have an entry either for Content1, hence, forwards the Consume Request upwards to CRME 1.1.
- Step 3: CRME 1.1 receives the Consume request for Content1. CRME 1.1 has an entry for Content1, which points to CRME 7.1. Hence, it forwards the Consume request downwards towards CRME 4.1, which, in turn, forwards it to CRME 7.1. Note that the Consume message does not follow peering links in this experiment.

Test 5: Basic State Installation in Topology 1 (i.e., when two CAFEs per domain are present).

- Step 1: State Installation messages are sent by the CRMEs to CAFEs
- Step 2: ContentTables are configured in CAFEs, after the State Installation messages are received. That means, for Topology 1, that CAFE 2.2 points to CAFE 2.3 and CAFE 2.3 to CAFE 4.2.

Test 6: Basic State Installation in Topology 2 (i.e., when more than two CAFEs are present per domain).

- Step 1: The same steps as in Test 5 are followed, that is, State Installation messages are sent from CRMEs to CAFEs.
- Step 2: ContentTables are configured in the respective CAFEs. In our case (Topology 2), CAFE 1.2 points to CAFE 2.2, while CAFE 1.4 points to CAFE 1.2. Note that in this case, no state is installed in CAFE 1.3.

Test 7: Basic content delivery in Topology 1.

- Step 1: After the Consume request has been received at CRME 5.1, the content is forwarded from CAFE 5.3 to CAFE 5.2. The content follows the route upwards towards CAFEs 1.3 and 1.2 and then downwards towards CAFEs 4.2 and 4.3
- Step 2: Content message is received by CAFE 4.3 and is saved locally at the content client.

Test 8: Content delivery in case of more than one CAFE per domain (Topology 2).

- Step 1: Similar to the previous experiment the content message is forwarded upwards from CAFE 7.3 and follows the Content State Installation in the CAFEs along the path to finally reach the Tier-1 domain.
- Step 2: Once it has reached the Tier-1 domain, the Content message follows the downward route towards CAFE 5.3 and the content client, where it is saved locally.

Test 9: Basic route optimisation process (Topology 2).

- Step 1/2: At each domain, the ingress CAFE of a content flow will notify its local CRME of new content flowing into its network. When this occurs between CAFE 2.2 and CRME 2.1, although a peering link exists between CAFE 2.2 and 3.2, this is not a valid BGP path for content to flow along, therefore, a lookup by CRME 2.1 in it routing table will yield no more optimal path than the current path, hence no further action will be taken.
- Step 3/4/5: In the case of the content flowing to the content consumer attached to CAFE 6.3, a more optimal route to the Content Server does exist. Therefore, when CRME 6.1 receives a Notify message from CAFE 6.2, it will perform a lookup in its routing table and when it finds a more optimal route to the Content Server, it will send a Consume message for the same content to the next-hop CRME along the more optimal route.
- Step 6: When CRME 7.1 receives this Consume message, it will configure the states in the ingress and egress CAFEs such that the requested content is sent from its domain directly to the domain with prefix 6, along the peering link.

Table 14: Overall validation of coupled approach operations.

Test Category	Test	TimeStamp/ Owner	Test Description	Steps	Output / Result	Result
1. Content Publication	1	06-02-2012/georgekamel	Publication of a single content item (topology 1)	Forwarding of Register message for MyContent1 from CP at 5.3:4000 to CRME at 5.1:3000	The following content tables are installed in the CRMEs at 5.1, 3.1 and 1.1, respectively: ContentTable at 5.1: Content ID | Next Hop Address | Next Hop Port MyContent1 5.3 4000 ContentTable at 3.1: Content ID | Next Hop Address | Next Hop Port MyContent1 5.1 3000 ContentTable at 1.1: Content ID | Next Hop Address | Next Hop Port MyContent1 3.1 3000	PASS
				Receipt, processing and generation of Publish message for MyContent1 at CRME at 5.1		
				Forwarding of Publish message for MyContent1 from CRME at 5.1:3000 to all parent CRMEs		
	2	09-02-2012/georgekamel	Publication of multiple content items (topology 2)	Forwarding of Register message for MyContent1 from CP at 7.3:1000 to CRME at 7.1:3000	The following content tables are installed in the CRMEs at 7.1, 4.1 and 1.1, respectively:	PASS

Test Category	Test	TimeStamp/ Owner	Test Description	Steps	Output / Result	Result
				Receipt, processing and generation of Publish message for MyContent1 at CRME at 7.1	ContentTable at 7.1: Content ID | Next Hop Address | Next Hop Port MyContent2 7.3 4001 MyContent1 7.3 4000 ContentTable at 4.1: Content ID | Next Hop Address | Next Hop Port MyContent2 7.1 3000 MyContent1 7.1 3000 ContentTable at 1.1: Content ID | Next Hop Address | Next Hop Port MyContent2 4.1 3000 MyContent1 4.1 3000	
				Forwarding of Publish message for MyContent1 from CRME at 7.1:3000 to all parent CRMEs		
				Forwarding of Register message for MyContent2 from CP at 7.3:1001 to CRME at 7.1:3000		
				Receipt, processing and generation of Publish message for MyContent2 at CRME at 7.1		
				Forwarding of Publish message for MyContent2 from CRME at 7.1:3000 to all parent CRMEs		
2. Content Resolution	**3**	06-02-2012/georgekamel	Request for non-existent content, MyContent2 (topology 1)	Forwarding of Consume message for MyContent2 from CC at 4.31000 to its local CRME at 4.1:3000	The following messages are given as output by the CRMEs at nodes 4.1, 2.1, and 1.1, respectively: CRME at 4.1: Consume packet for content MyContent2 received. ContentID MyContent2 not found in ContentTable. Consume message will be forwarded to a Provider CRME. CRME at 2.1: Consume packet for content MyContent2 received. ContentID MyContent2 not found in ContentTable. Consume message will be forwarded to a Provider CRME. CRME at 1.1 Consume packet for content MyContent2 received. Error: Content MyContent2 not found!	**PASS**
				Receipt of Consume message for MyContent2 by CRME at 4.1:3000, and forwarding of Consume message to all parent CRMEs		

Test Category	Test	TimeStamp/ Owner	Test Description	Steps	Output / Result	Result

	4	06-02-2012/georgekamel	Request for existent content item (topology 2)	Forwarding of Consume message for MyContent1 from CC at 5.3:1000 to its local CRME at 5.1:3000	The following messages are given as output by the CRMEs at nodes 4.1, 2.1, and 1.1, respectively:
					
					`CRME at 5.1:`
					`Consume packet for content MyContent1 received.`
					`ContentID MyContent1 not found in ContentTable.`
					`Consume message will be forwarded to a Provider CRME.`
					
					`CRME at 2.1:`
					`Consume packet for content MyContent1 received.`
					`ContentID MyContent2 not found in ContentTable.`
					`Consume message will be forwarded to a Provider CRME.`
					
					`CRME at 1.1`
					`Consume packet for content MyContent1 received.`
					`ContentTable contains entry for ContentID MyContent1`
					`Consume message will be forwarded to 4.1:3000`
					
					`CRME at 4.1`
					`Consume packet for content MyContent1 received.`
					`ContentTable contains entry for ContentID MyContent1`
					`Consume message will be forwarded to 7.1:3000`
					
					`CRME at 7.1`
					`Consume packet for content MyContent1 received.`
					`ContentTable contains entry for ContentID MyContent1`
					`Consume message will be forwarded to 7.1:4000`
					
					The CRME at 2.1 forwards the Consume message only to its parent CRME, and not to its peering CRME at 3.1
						PASS

				Receipt of Consume message for MyContent1 by CRME at 5.1:3000, and forwarding of Consume message to all parent CRMEs		
				Receipt of Consume message for MyContent1 by CRME at 1.1:3000, and forwarding of Consume message to all child CRMEs following states in content tables		

3. Content State Installation	5	07-02-2012/georgekamel	Basic state installation (topology 1)	Reception of Configure message by CAFEs from CRME	The following content state tables are installed in the domain with prefix 2:
					
					`CAFE ContentStateTable at 2.2:`
					`Content ID | Next Hop Address | Next Hop Port`
					`MyContent1 2.3 2000`
						PASS

Test Category	Test	TimeStamp/ Owner	Test Description	Steps	Output / Result	Result
				State installation in CAFE content state table	CAFE ContentStateTable at 2.3: Content ID | Next Hop Address | Next Hop Port MyContent1 4.2 2000 Similar content state tables are installed in domains with prefix 1, 3, 4, and 5.	
	6	09-02-2012/georgekamel	Basic state installation when more than two CAFEs are present within a single domain (topology 2)	Reception of Configure message by CAFEs from CRME State installation in CAFE content state table	The following content state tables are installed in the domain with prefix 1: CAFE ContentStateTable at 1.2: Content ID | Next Hop Address | Next Hop Port MyContent1 2.2 2000 CAFE ContentStateTable at 1.4: Content ID | Next Hop Address | Next Hop Port MyContent1 1.2 2000 No content state table is installed at CAFE 1.3.	PASS
4. Content Delivery	7	06-02-2012/georgekamel	Basic content delivery (topology 1)	Look up next hop from CAFE content state table Forward content to next hop	CAFE 5.3: Content MyContent1 received and will be sent to 5.2:2000 CAFE 5.2: Content MyContent1 received and will be sent to 3.3:2000 ... CAFE 4.2: Content MyContent1 received and will be sent to 4.3:2000 CAFE 4.3: Content MyContent1 received and will be sent to 4.3:1000 CC: Content MyContent1 received and saved to: ./Content/4.3 1000/MyContent1	PASS

Test Category	Test	TimeStamp/ Owner	Test Description	Steps	Output / Result	Result
	8	09-02-2012/georgekamel	Basic content delivery when more than two CAFEs are present within a single domain (topology 2)	Look up next hop from CAFE content state table Forward content to next hop	CAFE 7.3: Content MyContent1 received and will be sent to 7.2:2000 CAFE 7.2: Content MyContent1 received and will be sent to 4.3:2000 ... CAFE 5.2: Content MyContent1 received and will be sent to 5.3:2000 CAFE 5.3: Content MyContent1 received and will be sent to 5.3:1000 CC: Content MyContent1 received and saved to: ./Content/5.3 1000/MyContent1	**PASS**
	9	09-02-2012/georgekamel	Basic route optimisation process (topology 2)	CAFE 2.2 sends a `Notify` message to CRME 2.1 upon receipt of content Content1 CRME 2.1 performs a lookup in its routing table and determines that there is no shorter path to the Content Server and so takes no further action CAFE 6.2 sends a `Notify` message to CRME 6.1 upon receipt of content Content1 CRME 6.1 performs a lookup in *its* routing table and determines that there exists a more optimal path to the Content Server CRME 6.1 sends a Consume message for the same content directly to its peering domain	CAFE 2.2: Content MyContent1 received and will be sent to 2.3:2000 CRME 2.1: Notify message received from local CAFE 2.2 --- CAFE 6.2: Content MyContent1 received and will be sent to 6.3:2000 CRME 6.1: Notify message received from local CAFE 6.2 CRME 6.1: More optimal route for Content Content1	**PASS**

Test Category	Test	TimeStamp/ Owner	Test Description	Steps	Output / Result	Result
				CRME 7.1 receives this Consume message and re-configures the states in the ingress and egress CAFEs such that the requested content is sent from its domain directly to the domain with prefix 6, along the peering link	exists... ...sending Consume message to next hop CRME 7.1 along optimal route. CRME at 7.1: Consume packet for content MyContent1 received. CAFE ContentStateTable at 7.2: Content ID | Next Hop Address | Next Hop Port MyContent1 4.3 2000 MyContent1 6.2 2000 CAFE ContentStateTable at 7.3: Content ID | Next Hop Address | Next Hop Port MyContent1 7.2 2000	

4.5 Summary

This section presented in detail the Integration and Validation tests for the coupled approach that were carried out in the proof-of-concept prototype.

We started from the *validation of single entities* in Section 4.2, namely the CRME, CAFE, CP and CC. We have presented the steps taken by each entity throughout the publication, resolution, content state installation and delivery of the content, as well as the output from the proof-of-concept prototype.

The *overall validation* based on the publication, resolution and delivery and the steps taken by the entities during each process were discussed in section 4.4. In particular, in Table 2, we presented the detailed communication steps between the different entities in order to accomplish publication, resolution, content state installation and delivery of content from source to destination. In each of the steps, we have presented the output of the proof-of-concept prototype upon successful completion of the corresponding test.

5 Integration of applications

5.1 Introduction

In this chapter we focus on the integration of applications with the COMET system. Based on the analysis of use cases proposed in COMET deliverable D2.2 [1] and demonstration plans discussed in deliverable D6.1 [13], we have identified four types of applications that would be integrated with the COMET system. Each of these applications allows demonstrating specific content distribution mode supported by COMET. The selected applications cover:

1. Content streaming application. This application was selected to deal with live content, e.g. live transmission of sports or cultural events, TV or radio programs. This application assumes point-to-point communication with the content server.
2. Video on Demand application. This application was selected to consume pre-recorded content, like video on demand, music, etc. Basically, this application downloads the content or at least a piece of them and then it plays it out.
3. Peer-to-peer application. This application was selected to demonstrate capabilities of COMET system to offload the content servers by switching the consumer request to the p2p content distribution system. Such situation may happen when all content servers are fully loaded, e.g. in the case of flush crowds.
4. Content streaming rely application. This application was selected to demonstrate capabilities of COMET system to support point-to-multipoint connections. This application was specially designed for transmissions of live content, which is known a'priori to be very popular, e.g. opening ceremony of the Olympic Games, etc. In this case, the content streaming relay becomes additional source for this popular content.

In chapter 5.2, we present scenario how particular application should cooperate with COMET system, outline the necessary settings and configurations of COMET entities as well as we discuss required enhancements. Next, in chapter 5.3, we present validation tests and results that proved proper integration of considered applications.

5.2 Adaptation of applications

5.2.1 Streaming application

The basic scenario for integrating a streaming application in COMET is depicted in Figure 15, where the COMET entities involved in the integration are identified. The typical use case for the streaming application is a user that will request, via a browser, the retrieval of some content identified by a COMET Content Name and, after resolution, will receive a final URL to a streaming server, which will automatically cause the launching of the right player for the streaming protocol.

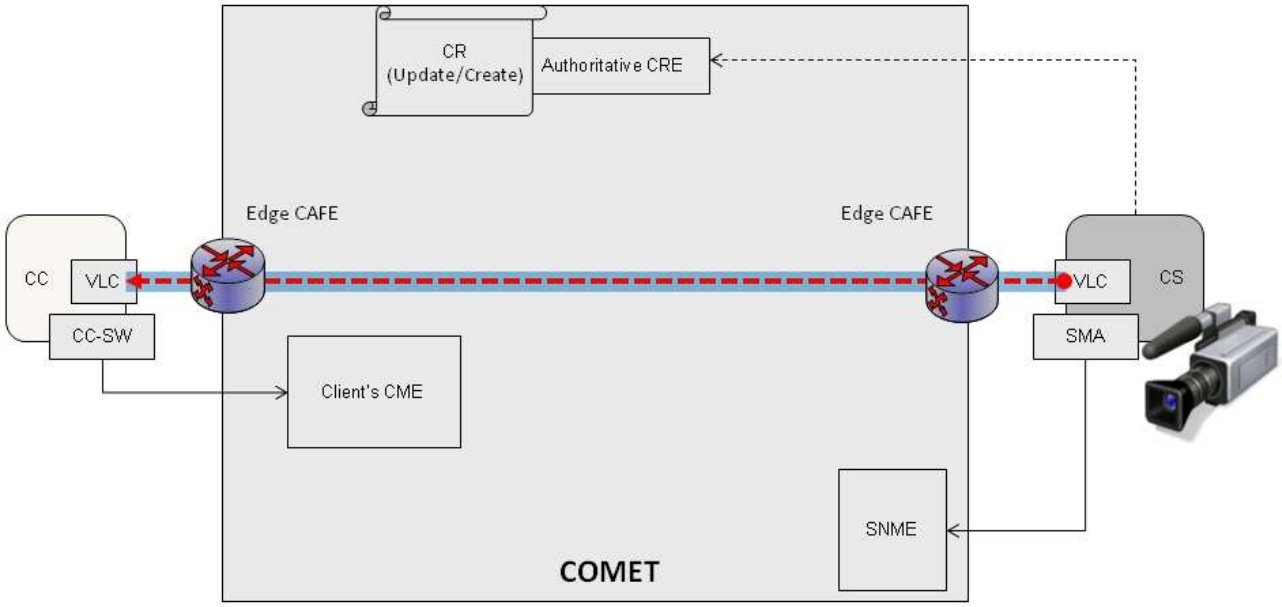

Figure 15: Basic COMET scenario for Streaming Applications

In this scenario, VLC [13] will be used as the streaming server in the CS and also as the Player in the Content Client. In the COMET prototype, two protocols will be used for streaming, HTTP, which in the current VLC implementation is supported for both IPv6 and IPv4, and RTSP, which is only supported for IPv4.

The way the streaming application (server and client) is integrated into COMET entities and the steps to be taken in every COMET entity for this integration are explained in the following subsections.

5.2.1.1 CS

In principle, COMET is designed to be deployed on existing CSs already providing service, without any changes on the way the service is being provided. The only action to be carried out in a real COMET environment is the installation of the SMA module that will enable the SNME to be informed about the CS status.

For emulating a Streaming Server in the COMET prototype, though, the selected software has been VLC. For its use as streaming the following conditions have to be taken into account when configuring the program

- For HTTP streaming, the URL the user has to request is of the type http://<server>:<port>/<content_name>.ts

  HTTP streaming can be used both for IPv6 and IPv4.

- For RTSP streaming, the URL the user has to request is of the type rtsp://<server>:<port>/<content_name> . VLC has to be specifically configured in order to:
  - Ensure that streaming only use a single flow, enabling CAFEs detection and control of the flow.
  - The port used by the Content Client for receiving the streaming will be fixed by the VLC server during the negotiation stage after connection.

  Owing to lack of support in the VLC current implementation, RTSP cannot be used over IPv6.

5.2.1.2 CC

The only requirement in the CC is the installation of the COMET CC SW which will act as a mediator between COMET and the browsers/players already installed in the user equipment. The COMET CC SW is an installable program (only implemented for Windows systems) that during installation will register the COMET CC SW as the application in charge of retrieving COMET URL (those beginning with comet://).

Once the COMET CC SW has been installed, the procedure for retrieving a video content is as follows:

- The user writes a COMET URL in the browser. For COMET demonstration purposes, the chosen browser is Firefox, but the procedure can be extended to any navigator.
- Firefox then hands over the URL to the COMET CC SW that will request the local CME for translation
- Once the COMET URL has been resolved, the COMET CC SW will first check if the application has a registry entry for the application protocol to be used for retrieval, if not, COMET CC SW will create an empty one.
- Once this check is performed, COMET CC will transfer the URL back to Firefox
- By using the protocol mapping to application in the registry, Firefox will launch the right application for retrieval or will prompt the user to choose one. For demonstration purposes, this application will be typically VLC.

5.2.1.3 CRE

In order to resolve the COMET Content Name properly into a final CS URL, the streaming server has to be registered in an authoritative CRE, either as a completely new Content Record or updating an already existing Content Record with a new CS in a Content Source.

The basic parameters to be defined in each Content Source are

- CoS: Either Premium or Best Effort
- Application protocol: Either HTTP or RTSP
- Transport protocol: Either TCP (for HTTP) or UDP (for RTSP)
- Port: server for HTTP, client for RTSP

And for each server

- Server IP
- Path to the content in the server for HTTP streaming.

5.2.1.4 SNME

The SIC modules in the SNME that receive the status of the CS in SNME have to be configured with the list of CS/SMA they are serving, so that the received statuses are taken into account. Only one SIC module is required to be configured as the entry point of the SMA commands.

5.2.1.5 CMEs

In order to integrate a new Content Client in COMET it has to be registered in the CME. This operation, carried out through the CME Administration tool, is simply the assignation of a COMET CoS (Premium, Better than Best Effort, Best Effort) to the new client

5.2.2 Video on demand application

The basic scenario for integrating a VoD application in COMET is depicted in Figure 16, where the COMET entities involved in the integration are identified. The typical use case for the VoD application is a user that will request, via a browser, the retrieval of some content identified by a COMET Content Name and, after resolution, will receive a final URL to a VoD server, which will automatically cause the launching of the right player for the VoD protocol.

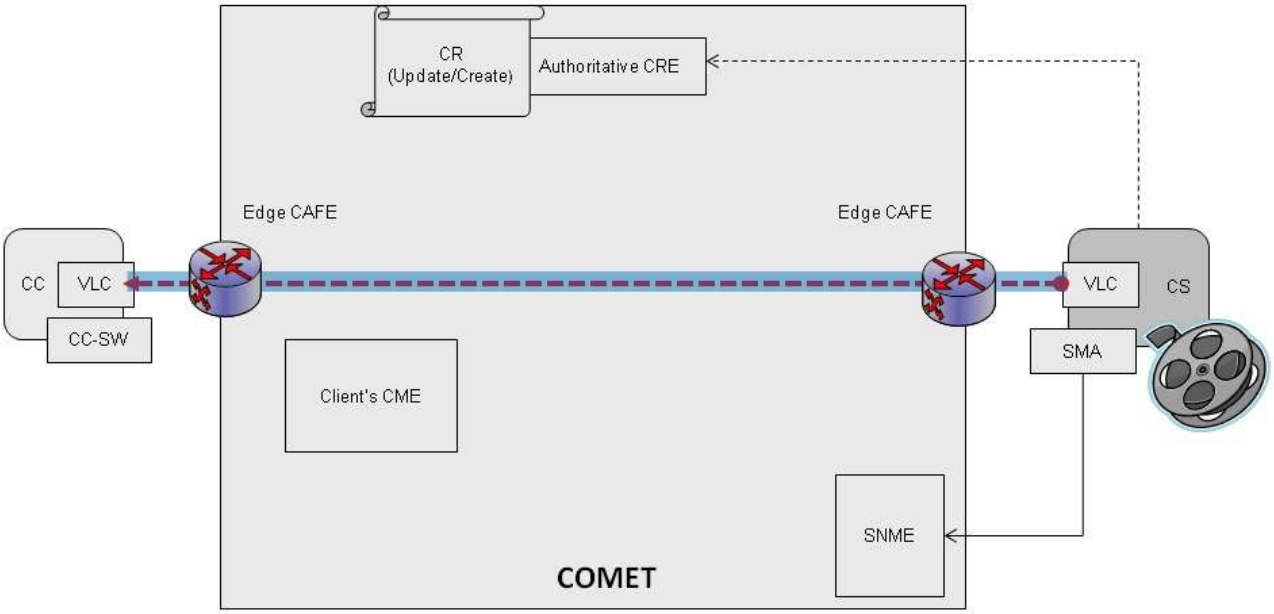

Figure 16: Basic COMET scenario for VoD Application

In this scenario, Apache Web Server [15] will be used as the VoD Server in the CS and VLC as the Player in the Content Client. This scenario is valid both for IPv4 and IPv6.

The way the streaming application (server and client) is integrated into COMET entities and the steps to be taken in every COMET entity are explained in the following subsections. Most of the integration measures are very similar to the Streaming Case, so only the differences will be described.

5.2.2.1 CS

COMET is designed to be deployed on existing CSs already providing service, without any changes on the way this service is being provided.

The only action to be carried out is the installation of the SMA module that will enable the SNME to be informed about the CS status.

For emulating a VoD Server in the COMET prototype, the chosen software has been Apache Web Server. The following conditions have to be taken into account when configuring the Apache server:

- Listening addresses/ports have to be configured according to the chosen IP version.
- The contents have to be stored in MPEG transport stream containers and the URLs accessing them must end in .ts

5.2.2.2 CC

As in the Streaming the only requirement in the CC is the installation of the COMET CC SW which will act as a mediator between COMET and the applications already installed in the user equipment. Same conditions and explanations than in the streaming case also apply here.

5.2.2.3 CRE

In order to resolve the COMET URLs properly into server URLs, the VoD server has to be registered in an authoritative CRE, either as a completely new Content Record or updating an already existing Content Record with a new CS in a Content Source.

The basic parameters to be defined in each Content Source are

- CoS: Either Premium or Best Effort
- Application protocol: HTTP
- Transport protocol: TCP
- Port: The one the Apache Server is listening

And for each server

- Server IP
- Path to the content in the server. It must conclude when a .ts extension

5.2.2.4 SNME

No difference with the Streaming Application

5.2.2.5 CMEs

No difference with the Streaming Application

5.2.3 P2p application

The basic scenario for integrating peer to peer (p2p) application with the COMET system is depicted in Figure 17, where the COMET entities involved in the integration are identified. This scenario is slightly different from the Streaming and VoD Application. In this case, only used for Best Effort COMET CoS, when the user requests the retrieval of the P2P content identified by the COMET Content Name, the resolution will returns an URL to a repository where a .torrent file with the tracker description is stored. Upon reception of this .torrent file the right application for P2P will be launched, but this process will fall outside the scope of COMET. Besides, since P2P traffic is classified as Best Effort in COMET, no paths end to end with assured QoS between both ends will be required.

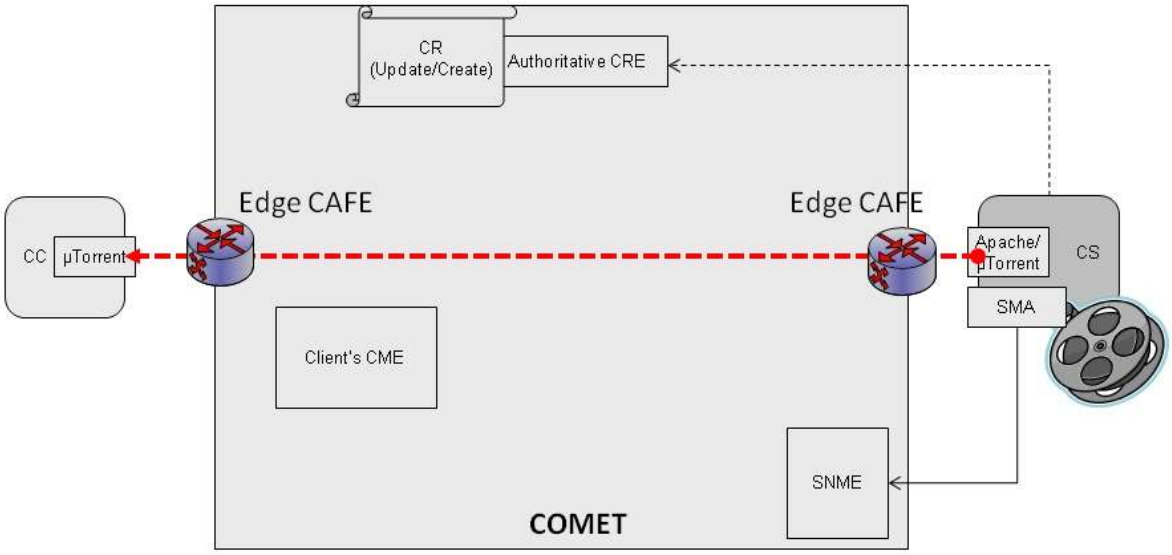

Figure 17: Basic COMET scenario for P2P Application

In this scenario, Apache [15] will be used as .torrent files repository in the CS, while µTorrent [16] will be also deployed in the CS in order to act as a torrent tracker and the first seed. In the Content Client, µTorrent will be also used as the P2P application for downloading the content

The way the P2P application (server and client) is integrated into COMET entities and the steps to be taken in every COMET entity are explained in the following subsections. Most of the integration measures are very similar to the Streaming and VoD Case, so only the differences will be described.

5.2.3.1 CS

COMET is designed to be deployed on existing CSs already providing service, without any changes on the way this service is being provided. Thus the only action to be carried out is the installation of the SMA module that will enable the SNME to be informed about the CS status.

In the particular case of the Apache as .torrent files repository in COMET the following conditions have to be taken into account when configuring the web server:

- Listening addresses/ports have to be configured according to the chosen IP version.

5.2.3.2 CC

As in the Streaming/VoD applications, the only requirement in the CC is the installation of the COMET CC SW, which will act as a mediator between COMET and the applications already installed in the user equipment. Same conditions and explanations than in the Streaming/VoD case also apply here.

5.2.3.3 CRE

In order to resolve the COMET URLs properly into server URLs, the P2P server has to be registered in an authoritative CRE, either as a completely new Content Record or updating an already existing Content Record with a new CS in a Content Source.

The basic parameters to be defined in each Content Source are:

- CoS: Only Best Effort.
- Application protocol: HTTP.
- Transport protocol: TCP.
- Port: The one the Apache Server is listening.

And for each server:

- Server IP.
- Path to the .torrent file with the description of the file.

5.2.3.4 SNME

No difference with the Streaming or VoD Application

5.2.3.5 CMEs

No difference with the Streaming or VoD Application

5.2.4 Content streaming relay application

The basic scenario for integrating the Content Streaming Relay (CSR) application with the COMET system is depicted in Figure 18, where the COMET entities involved in the integration are identified. The CSR application is used for supporting point-to-multipoint connections, and consists of two elements: Content Streaming Server (CSS) and Content Streaming Relay (CSR). The typical use case for this application involves two phases. The first phase is done before the content consumption and assumes setting the CSS and CSR for a new content that is known a'priori to be very popular. As a result of this step, the CSR becomes additional source of for this popular content. The second phase involves users that will request, via a browser, the retrieval of some content identified by a COMET Content Name. After resolution, the user will receive a final URL to the CSR, which will automatically cause the launching of the right player for the streaming protocol.

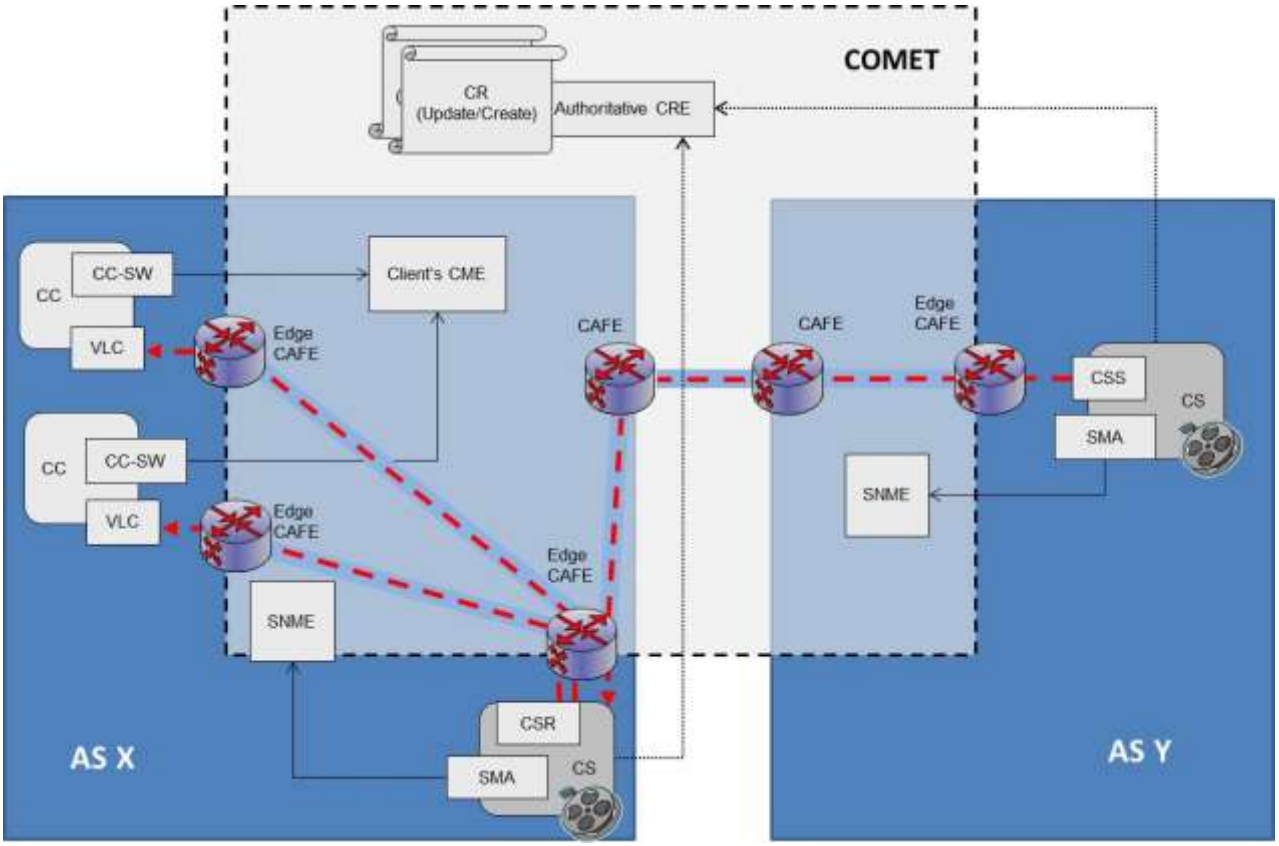

Figure 18: Basic COMET scenario for content streaming relay application

In this scenario, Tornado web server [17] is used as the streaming server in the CS and in the CSR. The VLC is used as the Player in the CC. In the COMET prototype HTTP protocol is used for content streaming made by CSS and CSR. The current implementation of Tornado web server and VLC client supports HTTP streaming for IPv4 and IPv6.

The way the content streaming relay application (CSS and CSR) and client application are integrated into COMET entities and the steps to be taken in every COMET entity for this integration are explained in the following subsections.

5.2.4.1 CS

The content streaming relay application consists of CSS and CSR. CSS and CSR are deployed on the existing CSs already providing service, with minimum changes on the way the service is being provided. The only requirements for the CSs are: 1) the installation of the SMA module that will enable the SNME to be informed about the CS status, 2) installation of COMET CSS module for CS with CSS function, or installation of COMET CSR module for CS with CSR function (see Figure 18).

- Notes for CSS

When configuring CC or CSR to connect to CSS following conditions have to be taken into account:

- for HTTP streaming over IPv4, the user requests the URL in the following form: http://<IPv4_address>:<port>/<content_name>.ts
- for HTTP streaming over IPv6, the user requests the URL in the following form: http://[<IPv6_address>]:<port>/<content_name>.ts

where IPv4/IPv6 addresses indicate address of CS containing CSS.

- Notes for CSR

When configuring CC to connect to CSR following conditions have to be taken into account:

- for HTTP streaming over IPv4, the user requests the URL in the following form: http://<IPv4_address>:<port>/<content_name>.ts
- for HTTP streaming over IPv6, the user requests the URL in the following form: http://[<IPv6_address>]:<port>/<content_name>.ts

where IPv4/IPv6 addresses indicate the address of CS containing CSR.

5.2.4.2 CC

No difference with the Streaming Application

5.2.4.3 CRE

- Notes for CSS

In order to resolve the COMET Content Name properly into the final CS URL, the CS with CSS function has to be registered in the authoritative CRE, either as a completely new Content Record or updating an already existing Content Record with a new CS in a Content Source.

- Notes for CSR

In order to resolve the COMET Content Name properly into the final CS URL, the CS with CSR function has to be registered in an authoritative CRE updating an already existing Content Record with a new CS in a Content Source.

5.2.4.4 SNME

No difference with the Streaming Application

5.2.4.5 CMEs

No difference with the Streaming Application

5.3 Validation tests

5.3.1 Streaming application

5.3.1.1 HTTP Streaming on IPv4

This test checks the integration with COMET of a HTTP streaming service on IPv4.

The testing procedure is composed of the following steps:

1. Install VLC, Firefox and the COMET CC SW on a CC with a Window OS and with an IPv4 address. The COMET Client has to be the last piece of software to be installed.
2. Install VLC and the SMA in a machine with Ubuntu 10 and an IPv4 address.
3. Launch the VLC in HTTP Streaming mode

   ```
   vlc my_content.mkv --sout
   '#std{access=http,mux=ts,dst=server_address_ipv4:port_server/path/file.ts}
   '
   ```

   where `my_content.mkv` is the file to be streamed `server_address_ipv4` is the IPv4 address of the server, `port_server` is the port in the server where connections are accepted and `path/file.ts` is the path to the content.
4. Create a CR in the CRE for serving the content. The CR has to contain at least:
   - In the Content Source section

        - CoS: Premium
        - Application protocol: HTTP
        - Transport protocol: TCP
        - Port: `port_server`
    - In the server section
        - Server IP: `server_address_ipv4`
        - Path: /`path/file.ts`.
5. Configure the remaining COMET entities, so that
    - SMA sends status information to the server's SNME
    - The CC is registered in the client's CME
6. Request the Content Name for the content in the CC's Firefox.

The expected result is that VLC will be automatically launched and the download of the content will start from the deployed CS, proving that HTTP streaming application over IPv4 can be integrated in COMET.

Actual Results: After deployment of a IPv4 CS as http content streaming server and COMET configuration, the COMET URL for that content was requested (i.e: comet://tid1.es/cleoSD) from a client equipment with COMET CC Software installed. The COMET URL was correctly resolved by the CME, VLC automatically launched and streaming started as though it had been directly requested by writing the content CS URL in the VLC prompt, i.e. http://10.95.51.13:8008/cleoSD.ts .

5.3.1.2 HTTP Streaming on IPv6

This test checks the integration with COMET of a HTTP streaming service on IPv6 .

The testing procedure is composed of the following steps:

1. Install VLC, Firefox and the COMET CC SW on a CC with a Window OS and an IPv6 address. The COMET Client has to be the last piece of software to be installed.
2. Install VLC and the SMA in a machine with Ubuntu 10 and an IPv6 address
3. Launch the VLC in HTTP Streaming mode.

   ```
   vlc my_content.mkv --sout --ipv6
   '#std{access=http,mux=ts,dst=[server_address_ipv6]:port_server/path/file.t
   s}'
   ```

   where `my_content.mkv` is the file to be streamed `server_address_ipv6` is the IPv6 address of the server, `port_server` is the port in the server where connections are accepted and `path/file.ts` is the path to the content.
4. Create a CR in the CRE for serving the content. The CR has to contain at least
    - For the Content Source
        - CoS: Premium
        - Application protocol: HTTP
        - Transport protocol: TCP
        - Port: `port_server`
    - For the server
        - Server IP: `server_address_ipv6`
        - Path: /`path/file.ts`.
5. Configure the remaining COMET entities, so that
    - SMA sends status information to the server's SNME
    - The CC is registered in the client's CME
    - CC and CS are attached to their respective edge CAFEs
6. Request the Content Name for the content in the CC's Firefox.

The expected result is that VLC will be launched and the download of the content will start from the deployed CS, proving that HTTP streaming application over IPv6 can be integrated in COMET.

Actual Results: After deployment of a IPv6 CS as http content streaming server and COMET configuration, the COMET URL for that content was requested (i.e: comet://tid1.es/cleoSD) from a client equipment with COMET CC Software installed. The COMET URL was correctly resolved by the CME, VLC automatically launched and streaming started as though it had been directly requested by writing the content CS URL in the VLC prompt, i.e., http://[2a02:9008:0:1912:0:50:56a3:48]:8008/cleoSD.ts .

5.3.1.3 RTSP Streaming on IPv4

This test checks the integration with COMET of a RTSP streaming service on IPv4.

The testing procedure is composed of the following steps:

1. Install VLC, Firefox and the COMET CC SW on a CC with a Window OS and an IPv4 address. The COMET Client has to be the last piece of software to be installed.
2. Install VLC and the SMA in a machine with Ubuntu 10 and an IPv4 address
3. Launch the VLC in HTTP Streaming mode

   `vlc my_content.mkv --sout '#rtp{dst=0.0.0.0,port=port,mux=ts,rtcp-mux=1,sdp=rtsp://:port/content_path}'`
   where `my_content.mkv` is the file to be streamed, `port` is both the port in the server where connections are accepted and the port in the client where the streaming is going to be received, while `content_path` is the path to the content.
4. Create a CR in the CRE for serving the content. The CR has to contain at least
   - For the Content Source:
     - CoS: Premium
     - Application protocol: RTSP
     - Transport protocol: UDP
     - Port: `port`
   - For the server
     - Server IP: Server Address
     - Path: /`content_path`
5. Configure the remaining COMET entities, so that
   - SMA sends status information to the server's SNME
   - The CC is registered in the client's CME
6. Request the Content Name for the content in the CC's Firefox.

The expected result is that VLC will be launched and the download of the content will start from the deployed CS, proving that RTSP streaming application over IPv4 can be integrated in COMET.

Actual Results: After deployment of a IPv4 CS as rtsp content streaming server and COMET configuration, the COMET URL for that content was requested (i.e: comet://tid1.es/cleoSD) from a client equipment with COMET CC Software installed. The COMET URL was correctly resolved by the CME, VLC automatically launched and streaming started as though it had been directly requested by writing the content CS URL in the VLC prompt, i.e., rtsp://10.95.51.13:5544/cleoSD.ts .

5.3.2 VOD application

5.3.2.1 VoD on IPv4

This test checks the integration with COMET of a VoD (HTTP) service on IPv4.

The testing procedure is composed of the following steps:

1. Install VLC, Firefox and the COMET CC SW on a CC with a Window OS and an IPv4 address. The COMET Client has to be the last piece of software to be installed.
2. Install Apache Web Server and the SMA in a machine with Ubuntu 10 and an IPv4 address
3. Configure the Apache Web server to receive queries on the port 80, by inserting the Line

   `Listen 80`

   in the `ports.conf` file at `/etc/apache2`
4. Deploy a file (like content.ts) at /var/www.The video content must be in MPEG Stream format and end with the ts extension
5. Create a CR in the CRE for serving the content. The CR has to contain at least
   - For the Content Source:
     - CoS: Premium
     - Application protocol: HTTP
     - Transport protocol: TCP
     - Port: 80
   - For the server
     - Server IP: IPv4 address of the CS
     - Path: `/content.ts`
6. Configure the remaining COMET entities, so that
   - SMA sends status information to the server's SNME
   - The CC is registered in the client's CME
7. Request the Content Name for the video content in the CC's Firefox.

The expected result is that VLC will be launched and the download of the content will start from the deployed CS, proving that VoD application over IPv4 can be integrated in COMET.

Actual Results: After deployment of an IPv4 CS as VoD server and COMET configuration, the COMET URL for that content was requested (i.e: comet://tid1.es/cleoSD) from a client equipment with COMET CC Software installed. The COMET URL was correctly resolved by the CME, VLC automatically launched and playing started as though it had been directly requested by writing the content CS URL in the VLC prompt, i.e., http://10.95.51.13/cleoSD.ts .

5.3.2.2 VoD on IPv6

This test checks the integration with COMET of a VoD (HTTP) service on IPv6.

The testing procedure is composed of the following steps:

1. Install VLC, Firefox and the COMET Client SW on a CC with a Window OS and an IPv6 address. The COMET Client has to be the last piece of software to be installed.
2. Install Apache Web Server and the SMA in a machine with Ubuntu 10 and an IPv6 address
3. Configure the Apache Web server to receive queries on the port 80 for IPv6 addressed, by inserting the Line

   `Listen [::]:80`

   In the `ports.conf` file at `/etc/apache2`
4. Deploy a file (like content.ts) at /var/www. The video content must be in MPEG Stream format and end with the ts extension
5. Create a CR in the CRE for serving the content. The CR has to contain at least
   - For the Content Source
     - CoS: Premium
     - Application protocol: HTTP

- Transport protocol: TCP
- Port: 80

- For the server
  - Server IP: IPv6 address of the CS
  - Path: `/content.ts`

6. Configure the remaining COMET entities, so that
   - SMA sends status information to the server's SNME
   - The CC is registered in the client's CME
7. Request the Content Name for the video content in the CC's Firefox.

The expected result is that VLC will be launched and the download of the content will start from the deployed CS, proving that VoD application over IPv4 can be integrated in COMET.

Actual Results: After deployment of a IPv6 CS as VoD server and COMET configuration, the COMET URL for that content was requested (i.e: comet://tid1.es/cleoSD) in a client equipment with COMET CC Software installed. The COMET URL was correctly resolved by the CME, VLC automatically launched and streaming started as though it had been directly requested by writing the content CS URL in the VLC prompt, i.e., http://[2a02:9008:0:1912:0:50:56a3:48]/cleoSD.ts .

5.3.3 P2P application

5.3.3.1 P2P on IPv4

This tests checks the integration with COMET of a P2P service on IPv4.

The testing procedure is composed of the following steps:

1. Install µTorrent, Firefox and the COMET Client SW on a CC with a Window OS and an IPv4 address. The COMET Client has to be the last piece of software to be installed.
2. Install Apache Web Server, µTorrent and the SMA in a machine with Windows OS and an IPv4 address
3. Configure the Apache Web server to receive queries on the port 80 for IPv4, by inserting the Line

   `Listen 80`

   In the `httpd.conf` file at `/etc/apache2` at `/ C:\Program Files\Apache Software Foundation\Apache2.2\htdocs` or `:\Program Files (x86)\Apache Software Foundation\Apache2.2\htdocs`
4. Configure µTorrent for tracking and create a content.torrent file for the video content as explained in section 13.3.3
5. Deploy the content.torrent file at C:\Program Files (x86)\Apache Software `Foundation\Apache2.2\htdocs` or `C:\Program Files\Apache Software Foundation\Apache2.2\htdocs`, so that it is accessible with the URL http://server-IPv4/content.torrent
6. Create a CR in the CRE for serving the content. The CR has to contain at least
   - The basic parameters to be defined in each Content Source are
     - CoS: Premium
     - Application protocol: HTTP
     - Transport protocol: TCP
     - Port: 80
   - And for each server
     - Server IP: IPv4 address of the CS
     - Path: `/content.torrent`
7. Configure the remaining COMET entities, so that

- SMA sends status information to the server's SNME
- The CC is registered in the client's CME

8. Request the Content Name for the video content in the CC's Firefox.

The expected result is that µTorrent will be launched and the download of the content will start from the peer overlay, proving that P2P applications over IPv4 can be integrated in COMET.

Actual Results: After deployment of a CS as P2P tracker and COMET configuration, the COMET URL for that content was requested (i.e: comet://tid1.es/cleoSD) in a client equipment with COMET CC Software installed. The COMET URL was correctly resolved by the CME, µTorrent automatically launched and download started as though it had been directly requested by writing the .torrent URL in the Firefox prompt, i.e., http://10.95.51.36:8080/cleoSD.torrent.

5.3.3.2 P2P on IPv6

This test checks the integration of p2p application running on IPv6 with COMET.

The testing procedure is composed of the following steps:

1. Install µTorrent, Firefox and the COMET Client SW on a CC with a Window OS and with an IPv6 address. The COMET Client has to be the last piece of software to be installed.
2. Install Apache Web Server, µTorrent and the SMA in a machine with Windows OS and an IPv6 address
3. Configure the Apache Web server to receive queries on the port 80 for IPv6, by inserting the Line

   ```
   Listen [::]:80
   ```

   In the `httpd.conf` file at `/etc/apache2` at `/ C:\Program Files\Apache Software Foundation\Apache2.2\htdocs` or `:\Program Files (x86)\Apache Software Foundation\Apache2.2\htdocs`
4. Configure µTorrent for tracking and create content.torrent file for the video content as explained in in section 13.3.3
5. Deploy the content.torrent file at `C:\Program Files (x86)\Apache Software Foundation\Apache2.2\htdocs` or `C:\Program Files\Apache Software Foundation\Apache2.2\htdocs`, so that it is accessible with the URL http://[server-IPv6]/content.torrent
6. Create a CR in the CRE for serving the content. The CR has to contain at least
   - For the Content Source
     - CoS: Premium
     - Application protocol: HTTP
     - Transport protocol: TCP
     - Port: 80
   - For the server
     - Server IP: IPv6 address of the CS
     - Path: `/content.torrent`
7. Configure the remaining COMET entities, so that
   - SMA sends status information to the server's SNME
   - The CC is registered in the client's CME
8. Request the Content Name for the video content in the CC's Firefox.

The expected result is that µTorrent will be launched and the download of the content will start from the peer overlay, proving that P2P applications over IPv6 can be integrated in COMET.

Actual Results: After deployment of a CS as P2P tracker and COMET configuration, COMET URL for that content was requested (i.e: comet://tid1.es/cleoSD) in a client equipment with COMET CC Software installed. The COMET URL was correctly resolved by the CME, µTorrent automatically launched and download started as though it had been directly requested by writing the .torrent URL in the Firefox prompt, i.e. http://[2a02:9008:0:1912:0:50:56a3:48]:8080/cleoSD.torrent

5.3.4 Content streaming relay application

5.3.4.1 HTTP point-to-multipoint streaming on IPv4

This test checks, the integration COMET with the point-to-multipoint HTTP streaming service on IPv4.

The testing procedure is composed of the following steps:

1. Install VLC, Firefox and the COMET CC SW on two CCs with a Window OS and with an IPv4 address. The COMET Client has to be the last piece of software to be installed. As a result of this step we have two CCs (say CC_1 and CC_2) ready for contentment consumption
2. Install the CSS and the SMA in a machine with Ubuntu 10 and an IPv4 address. This machine is used as CS with CSS function. This machine should not be located in the AS where COMET CC is located. As a result of this step we have one CS with CSS module (say CS_1).
3. Install CSR and the SMA in a machine with Ubuntu 10 and an IPv4 address. This machine is used as CS with CSR function. This machine should be located in the AS where COMET CC is located. As a result of this step we have one CS with CSR module (say CS_2).
4. Launch the CSS (in CS_1) in HTTP Streaming mode with following command:

   `"python server.py -r rate -p port_CSS path/file"`
   where `rate` is the rate of the stream, `port_CSS` is the port in the server where connections are accepted and `path/file` is the path to the content.
5. Create a CR in the CRE for serving the content. The CR has to contain at least:
   - In the Content Source section
     - CoS: Premium
     - Application protocol: HTTP
     - Transport protocol: TCP
     - Port: `port_CSS`
   - In the server section
     - Server IP: `IPv4 address of CS containing CSS (CSS_IP_address)`
     - Path: /`path/file`
6. Configure the remaining COMET entities, so that
   - SMA sends status information to the server's SNME
   - The CC is registered in the client's CME
7. Request the Content Name for the content in the CC_1's Firefox.
8. Measure the comet traffic load in border CAFE at CS_1 side. Denote result of this measurement as $result_A$.
9. Request the Content Name for the content in the CC_2's Firefox
10. Measure the comet traffic load in border CAFE at CS_1 side. Denote result of this measurement as $result_B$.
11. Stop content consumptions from CC_1 and CC_2.
12. Launch the CSR (in CS_2) in HTTP Streaming mode with following command:

    `"python relay.py -p port_CSR http://CSS_IP_address:port_CSS/stream.ts"`

    where `port_CSR` is the port in the server where connections are accepted, `CSS_IP_address` is the IPv4 address of CS containing CSS, `port_CSS` is the port number on which CSS is running.
13. Update a CR in the CRE for with new content source. The CR has to contain at least:
    - In the server section

- Server IP: `IPv4 address of CS containing CSS (CSS_IP_address)`
- Path: /`path/file`

14. Configure the remaining COMET entities, so that
    - SMA sends status information to the server's SNME.
15. Request the Content Name for the content in the CC_1's Firefox.
16. Measure the comet traffic load in border CAFE at CS_1 side. Denote result of this measurement as $result_C$.
17. Request the Content Name for the content in the CC_2's Firefox.
18. Measure the comet traffic load in border CAFE at CS_1 side. Denote result of this measurement as $result_D$.

The expected result are as follows:

- the VLC is automatically launched at each content request and HTTP streaming over IPv4 takes place,
- the traffic loads measured at border CAFE at CS_1 side fulfill following formula: $result_A = result_C = result_D = result_B$. This mean that CSR is performing multicast as supposed.

Since obtained results were the same as expected we conclude that content streaming relay application works correctly over IPv4.

5.3.4.2 HTTP point-to-multipoint streaming on IPv6

This test checks, the integration COMET with the point-to-multipoint HTTP streaming service on IPv6.

The testing procedure is composed of the following steps:

1. Install VLC, Firefox and the COMET CC SW on two CCs with a Window OS and with an IPv6 address. The COMET Client has to be the last piece of software to be installed. As a result of this step we have two CCs (say CC_1 and CC_2) ready for contentment consumption
2. Install Tornado web server, CSS and the SMA in a machine with Ubuntu 10 and an IPv6 address. This machine is used as CS with CSS function. This machine should not be located in the AS where COMET CC is located. As a result of this step we have one CS with CSS module (say CS_1).
3. Install Tornado web server, CSR and the SMA in a machine with Ubuntu 10 and an IPv6 address. This machine is used as CS with CSR function. This machine should be located in the AS where COMET CC is located. As a result of this step we have one CS with CSR module (say CS_2).
4. Launch the CSS (in CS_1) in HTTP Streaming mode with following command:

   `"python server.py -r rate -p port_CSS path/file"`
   where `rate` is the rate of the stream, `port_CSS` is the port in the server where connections are accepted and `path/file` is the path to the content.
5. Create a CR in the CRE for serving the content. The CR has to contain at least:
   - In the Content Source section
     - CoS: Premium
     - Application protocol: HTTP
     - Transport protocol: TCP

        - Port: `port_CSS`
    - In the server section
        - Server IP: `IPv6 address of CS containing CSS (CSS_IP_address)`
        - Path: /`path/file`
6. Configure the remaining COMET entities, so that
    - SMA sends status information to the server's SNME,
    - The CC is registered in the client's CME.
7. Request the Content Name for the content in the CC_1's Firefox.
8. Measure the comet traffic load in border CAFE at CS_1 side. Denote result of this measurement as $result_A$.
9. Request the Content Name for the content in the CC_2's Firefox
10. Measure the comet traffic load in border CAFE at CS_1 side. Denote result of this measurement as $result_B$.
11. Stop content consumptions from CC_1 and CC_2.
12. Launch the CSR (in CS_2) in HTTP Streaming mode with following command:

    `"python relay.py -p port_CSR http://[CSS_IP_address]:port_CSS/stream.ts"`

    where `port_CSR` is the port in the server where connections are accepted, `CSS_IP_address` is the IPv4 address of CS containing CSS, `port_CSS` is the port number on which CSS is running.
13. Update a CR in the CRE for with new content source. The CR has to contain at least:
    - In the server section
        - Server IP: `IPv6 address of CS containing CSS (CSS_IP_address)`
        - Path: /`path/file`
14. Configure the remaining COMET entities, so that
    - SMA sends status information to the server's SNME.
15. Request the Content Name for the content in the CC_1's Firefox.
16. Measure the comet traffic load in border CAFE at CS_1 side. Denote result of this measurement as $result_C$.
17. Request the Content Name for the content in the CC_2's Firefox.
18. Measure the comet traffic load in border CAFE at CS_1 side. Denote result of this measurement as $result_D$.

The expected result are as follows:

- the VLC is automatically launched at each content request and HTTP streaming over IPv4 takes place,
- the traffic loads measured at border CAFE at CS_1 side fulfill following formula: $result_A = result_C = result_D = result_B$. This mean that CSR is performing point-to-multipoint streaming as supposed.

Since obtained results were the same as expected we conclude that content streaming relay application works correctly over IPv6.

6 Performance metrics

This chapter presents the set of performance metrics that were identified to assess performance of the COMET system. The proposed metrics are related to the overall COMET system performance as well as the benchmarks for evaluation of the major COMET processes. In particular, the proposed metrics focus on the performance of: (1) overall content resolution and retrieval process, (2) content publication process, (3) name resolution process, (4) server and network awareness process, (5) routing awareness process, and (6) content forwarding process. The presented performance metrics would constitute a base for performance evaluation of COMET prototype, assessment of COMET system scalability and comparison of COMET approaches. As far as possible, we have identified the reference values for each metric based on analysis of related works or analysis of currently exploited reference systems.

6.1 Metrics related to content retrieval

6.1.1 Content Retrieval Latency and Content Resolution Time

6.1.1.1 Definition

Content Retrieval Latency (CRL) is an overall metric, which reflects how long a consumer waits for content delivery. The CRL is defined as the time interval between the moment when a consumer sends the request message to the COMET system and the moment when the consumer gets the response, either the first piece of the requested content or negative response. In the case of the decoupled approach, the CRL covers two major components that are: (1) Content Resolution Time (CRT) and (2) Content Server Response Delay (CSRD). The CRT time elapses between the moment when a consumer sends the content request to the COMET system and receives the URL of the content server which was selected by COMET to serve consumer request. The CRT directly relates to the performance of the COMET decoupled approach where content resolution and consumption phases are separated. The CRT composes of a number of constituent latencies related to: (1) name resolution process, (2) collecting server and network information and (3) cafe configuration. The CSRD is the content server response delay, which elapses between the moment when the consumer application sends a request command to the selected content server and when it receives the first piece of the content. Note that CSRD mostly depends on the processing time required at the content server and the network round trip time. These delay components are basically out of the COMET control.

On the other hand, in the COMET coupled approach both processes of content resolution and content server communication are merged together. Therefore, in this case, we cannot distinguish CRT and CSRD delay components, but consider a single overall CRL metric. However, this delay may be considered as a number of constituent latencies: (1) CRME content table lookup time, which will in turn be dependent on the specific lookup algorithm being used as well as the number of content records contained in the CRME's content table; and (2) AS-level transfer delay, which can approximated by using the measured hop count and the equations given by Rajajalme *et al* in [18].

6.1.1.2 Reference values

The CRL is an overall metric, which composes of resolution delay introduced by the COMET system, the consumer and server sides processing delay and the transfer delay introduced by the network. In order to assess the reference values for the CRL metric, we considered two reference content delivery scenarios. The first scenario refers to the currently exploited CDNs. These systems reduce content retrieval latency by placing content servers and resolution mechanism close to the consumers. From the COMET point of view, this scenario corresponds to the situation where consumers, COMET servers and content itself are located close together, e.g. in a single domain.

On the other hand, the COMET was designed to provide global, unified access for any content available all over the world. In particular, for contents located far away from the consumer. In such situation, the content resolution time will be longer mainly due to large round trip delays. In order to assess the upper limit for CRL we considered user's Tolerable Waiting Time (TWT) studied for web page downloading, e.g. in [20]. Although the TWT metric is not strictly related to the multimedia content consumption, we believe it can approximate user's tolerance for delay in content retrieval.

For the CDN scenario, we identified the reference CRL value based on the analysis of related works focusing on CDN performance. The most representative seems [19], where authors measured the performance of two largest CDNs. In Table 15, we present their results showing the 95 percentile of name resolution time and server response delay measured in AKAMAI and LIMELIGHT networks.

Table 15: 95 percentile of name resolution time and server response delay in the largest CDNs [19].

	Akamai		Limelight	
delay (ms)	DNS	server	DNS	server
North America	115.81	67.24	78.64	79.03
Europe	131.08	82.54	103.34	110.05
Asia	122.5	125.53	199.84	284.4
Oceania	163.68	173.17	279.12	266.66
South America	402.35	312.52	388.17	368.66
Africa	961.03	647.08	596.03	591.45

Source: [19] Ch. Huang, A. Wang, J. Li, K. W. Ross, Measuring and evaluating large-scale CDNs, 8th ACM SIGCOMM conference on Internet measurement (IMC '08)

Following the CDN performance results, we assume that 95 percentile of CRL should be about 213 ms for a single domain scenario. This value is representative for both CDNs in Europe.

On the other hand, in order to define the upper bound of CRL, which is more adequate for the large scale networks, we focused on the user's tolerable waiting time. Following the survey presented in [20], the user's TWT has significantly changed during last decade. In Table 16, we present findings and recommendations from [20].

Table 16: Survey of user's tolerable waiting time presented in [20]

Study	Findings/Recommendations
Ramsey, Barbesi and Preece (1998)	• Delay of 41 seconds is suggested as the cut-off for long delays based on users' perceptions
Selvidge (1999)	• Delay of 30 seconds is suggested as the cut-off based on users' performance and frustration levels
Nielsen (1993, 1995, 1996)	• Delay of 15 seconds is tolerable in the Web context
Hoxmeier and DiCesare (2000)	• Delay of 12 seconds causes satisfaction to decrease
Galletta, Henry, McCoy and Polak (2002)	• Delay of 4 seconds causes performance and behavioural intentions to stabilize whereas attitudes remain unchanged after delay exceeds 8 seconds

Source: [20], Nah, F. (2004), A study on tolerable waiting time: how long are Web users willing to wait? Behaviour & Information Technology, vol. 23 issue 3, 2004

The authors conclude that while user's TWT was about 41 seconds in 1998, it became only 8 seconds in 2002. The decreasing tendency of the user's TWT is still observed. In particular, the latest recommendations for web page developers [23] suggest keeping web page download time below 3 or 4 seconds. Following conclusions from user's TWT studies, we assume that 95 percentile

of CRL should not exceed 3 s. This value would constitute an upper bound for CRL, which we believe is tolerable by users.

In the COMET decoupled approach, we distinguish the Content Resolution Time (CRT) as a component of content retrieval latency. The CRT defines the time required by the COMET system to resolve content name and prepare the network for content delivery. From the functional point of view, the COMET resolution and DNS resolution processes may be treated as equivalent. In principle, both processes translate the content name, i.e. the content ID or the server name, into the IP address of the server.

Therefore, in order to assess the reference values for CRT, we also consider two reference systems, the DNS provided by CDNs and DNS available in the Internet. The first system refers to the COMET single domain scenario as CDNs provide own DNS servers that are located close to users. The second considered system is related to the large-scale scenario, where DNS system resolves requests from hosts located in distant locations.

Following CDN performance results summarised in Table 15, we assume that 95 percentile of CRT should not exceed 130 ms in a single domain scenario. On the other hand for large scale scenario, we analysed characteristics of DNS lookup latency measured in [24]. In Figure 19, we present the results reported in [24], which say that the 95 percentile of DNS lookup latency is about 2.5 s. This value correspond to the DNS requests served by the root DNS.

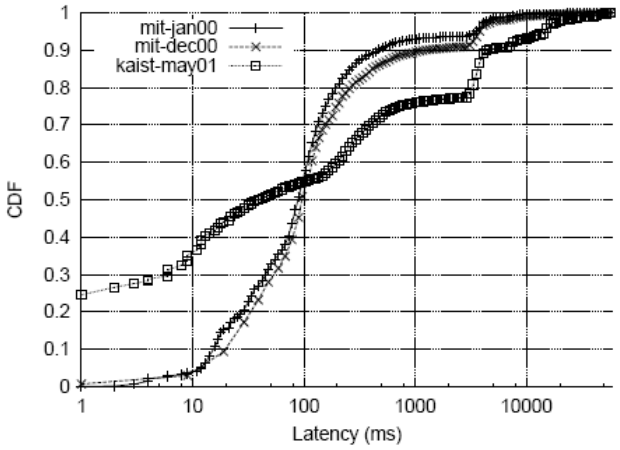

Figure 19: The DNS lookup latency reported in [24].

Source: Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris, DNS Performance and the Effectiveness of Caching, IEEE/ACM Transactions on Networking (TON) archive, Volume 10 Issue 5, October 2002 Pages 589 - 603

Based on the above analysis of the DNS lookup latency, we assume that 95 percentile of CRT should not exceed 2.5 s for the large-scale scenario. Note that 2.5 s limit for CRT is also consistent with the assumed tolerable limit for content retrieval latency.

6.1.2 Content Retrieval Success Ratio

6.1.2.1 Definition

Content Retrieval Success Ratio (CRSR) is defined as the ratio of the number successfully retrieved content to the number of retrieval attempts (requests) whenever the targeted content exists and there are available resources at network level and content servers. The content is treated as successfully retrieved when consumer receives the right content before the abandon timeout expires. This metric will be used to assess the efficiency of the COMET system working in the

federated testbed environment. Moreover, we use this metric to compare the effectiveness of COMET approach with other content delivery systems. Such studies will be performed by simulation.

6.1.2.2 Reference values

Under normal load conditions, the CRSR should not be lower than 99.9%

6.1.3 Content resolution signaling overhead

6.1.3.1 Definition

In the coupled approach, two resolution schemes are proposed to deal with the situation of multi-homed autonomous systems. The first is a *random* resolution scheme, whereby a multi-homed AS will randomly chose a single provider to which to forward the content resolution request message. The second is a broadcast resolution scheme whereby a multi-homed AS will send a content resolution request message to all of its providers. In order to evaluate these schemes, the total number of ASes traversed in attempting to resolve a content request is determined.

6.1.3.2 Reference values

We aim to compare overhead of the random and broadcast resolution modes, therefore, there are no reference values *per se*.

6.2 Metrics related to Content publication

6.2.1 Number of content records stored on CRE

6.2.1.1 Definition

The maximum number of content records stored in a single CRE (either the authoritative and root). This metrics will be used in scalability studies to assess how the system should evolve when the number of content records increases.

6.2.1.2 Reference values

The reference values are difficult to assess. Raw estimation would be done based on content record size, database overhead and available disk space. The minimum content record size is estimated to 620 bytes, while the overhead produced by the Berkeley database is around 170 bytes per record, resulting to a minimum of 790 bytes for the simplest content record. Hence, a single CRE with 1GB HDD would store around 1.26 million content records, or a 1TB HDD would store over a billion of content records.

6.2.2 Maximum number of users connected to the CP

6.2.2.1 Definition

The maximum number of users connected to a single CP. This metric expresses the CP ability to handle simultaneous connection from a number of users.

6.2.2.2 Reference values

The reference values are difficult to assess. Raw estimation would be done by comparison to HTTP server capabilities. The motivation comes from the fact that CP interface is based on HTTP protocol.

6.2.3 Maximum publication rate

6.2.3.1 Definition

The maximum publication rate is expressed in [pub/s]. This metric reflects the CP ability to handle simultaneous publication requests without dropping requests or significant degradation of the response time.

6.2.3.2 Reference values

The reference values are difficult to assess. Raw estimation would be done based on the maximum database query rate. Publication time is estimated around 10-30ms, hence the expected maximum publication rate would be around 30 - 100 pub/s.

6.3 Metrics related to CRE

6.3.1 Maximum query rate

6.3.1.1 Definition

The maximum query rate of both root and authoritative CRE is expressed in [req/s]. This metric reflects the ability of CRE to handle simultaneous resolution requests without dropping requests or significant degradation of the response time.

6.3.1.2 Reference values

The reference values are difficult to assess. Raw estimation of reference values for root CRE would be done based on root DNS performance characteristic, which is around 500 req/s in exploited networks. On the other hand, the expected CRE response time in a single host is estimated around 1-2 ms, hence the maximum query rate would be around 500-1000 req/s.

6.3.2 CRE response time

6.3.2.1 Definition

The response time of root or authoritative CRE is defined as the time interval between the moment when CME sends the request message to CRE and it receives the response.

6.3.2.2 Reference values

The reference values are difficult to assess. Raw estimation of reference values for CRE would be done based on the expected CRE response time in single host, which is around 1-2 ms. So, the estimated response time would be around 1-2 ms + RTT

6.4 Metrics related to SNME

6.4.1 Maximum query rate

6.4.1.1 Definition

This metric defines the maximum query rate of server status that can be handled by a single SNME. Since each query can carry several CSs status requests, response times will be measured against this parameter in order to assess possible means of optimisation. This metric will be used to express the performance of SNME entity as well as in scalability studies to assess how the system should evolve when the rate of content requests increases.

6.4.1.2 Reference values

The reference values are difficult to assess. In the worst case, the SNME should handle as many queries as content requests arriving at the CME.

6.4.2 SNME response time

6.4.2.1 Definition

Maximum tolerable SNME response time for CS status queries is expressed by 95% percentile of the time interval between the moment when CME sends the request message to SNME and it receives the response. Since each query can carry several CSs status requests, response times will be measured against this parameter in order to assess possible means of optimisation.

6.4.2.2 Reference values

Reference values are difficult to assess. In any case, response times should be much lower (at least 1 order of magnitude) than resolution times (CRT) for the entire system (see section 6.1.1), to avoid degradation of the entire system performance.

6.4.3 Number of CS served by a SIC-SNME

6.4.3.1 Definition

This metric express how many CSs can be managed by a single SIC module deployed in SNME, before deployment of a new SIC module is required. Three parameters, Memory Occupation, CPU load and SYN overhead in the SMA/SIC protocol will be monitored in the SNME in order to detect when new SIC modules are required.

6.4.3.2 Reference values

Reference values are difficult to assess. As a rule, a single SIC module is expected to be able to handle dozens of CS.

6.5 Metrics related to CME

6.5.1 Maximum request rate

6.5.1.1 Definition

The maximum number of requests per second that can be handled by CME in CC-CME and inter-CME interfaces is expressed in [req/s]. This metric reflects the ability of CME to handle CC requests or other CME requests without dropping them or significant degradation of the response time.

6.5.1.2 Reference values

The reference values are difficult to assess. This metric will be evaluated in the testbed and obtained results will be used in scalability studies to assess how the system should evolve when the number of requests increases.

6.5.2 Response time

6.5.2.1 Definition

The response time (Minimum/Mean/Maximum and 95 percentile) of the client-adjacent and server-adjacent CME in CC-CME and inter-CME interfaces respectively. It is defined as the time interval between the moment when CC (or other CME) sends the request message to CME and it receives the response.

6.5.2.2 Reference values

The reference values are difficult to assess. This metric will be evaluated in the testbed and obtained results will be used in scalability studies to assess how the system should evolve when the number of requests or round trip time increases. In any case, response times should be much lower (at least 1 order of magnitude) than resolution times (CRT) for the entire system (see section 6.1.1), to avoid degradation of the entire system performance.

6.6 Metrics related to RAE

6.6.1 Routing Convergence Time

6.6.1.1 Definition

The Routing Convergence Time (RCT) is one of the basic performance indicators related to routing protocols [25]. It defines the total amount of time that elapses between the occurrence of the routing stressing event, e.g. advertisement of a new prefix, withdrawal of existing one, link failure, and the time instant when the last update message caused by this event is processed by the Routing Awareness Entity (RAE) and, as a consequence, the routing is stable again.

6.6.1.2 Reference values

RCT strongly depends on the number of domains, network topology, processing power of routers, transmission delays, etc. Therefore, it is difficult to provide explicit reference value. However, in order to evaluate RAE performance, we compare the RCT measured for RAE with the RCT measured for BGP-4 routing protocol running in the same network.

6.6.2 Number of stored network prefixes

6.6.2.1 Definition

This metric defines the number of network prefixes that could be stored and processed by RAE. This metric evaluate the efficiency of RAE prototype under large number of network prefixes.

6.6.2.2 Reference values

The RAE should handle at least the same number of network prefixes as available in the current Internet, i.e. around 300 000 prefixes [26], [27], [28].

6.7 Metrics related to Content delivery

6.7.1 Lossless throughput of CAFE

6.7.1.1 Definition

The lossless throughput defines the maximum rate of packets carried by CAFE without packet losses, see RFC 5180 [29] for details. This metric will be used to evaluate the performance of CAFE.

6.7.1.2 Reference values

It is difficult to provide an explicit reference value as CAFE performance depends on the hardware it is running (especially the processing power, performance of network interfaces, etc.). Therefore, we compare the performance of CAFE with the software IP router. The performance of CAFE should be similar to the performance of software IP routers running on the same hardware.

6.7.2 Number of simultaneous flows handled by CAFE

6.7.2.1 Definition

This metric defines the number of simultaneous flows that could be configured and intercept by the edge CAFE. This metric is important to evaluate how many flows could be carried by a single edge CAFE.

6.7.2.2 Reference values

At 1 Gbps port, the maximum number of simultaneous flows should be on the order of 10^5. This value corresponds in average to 100 kbps per a single flow.

6.7.3 Configuration latency of edge CAFE

6.7.3.1 Definition

The edge CAFE configuration latency defines the time elapsing between the moment when the cafe_configurator sends the command to configure the packet filtering rule on the edge CAFE and the moment when it receives confirmation that requested filter has been properly configured.

6.7.3.2 Reference values

Taking into account that each flow handled by Premium or Better Then Best Effort Class of Service must by be configured at the edge CAFE (only one CAFE), the configuration latency should not exceed a fraction of a second in order to meet requirements for CRT defined in the section 6.1.1.

6.7.4 Size of Forwarding Information Base in CAFE

6.7.4.1 Definition

The size of Forwarding Information Base (FIB) defines the number of entries stored in the CAFE forwarding table. This metric has impact on the CAFE performance, especially the lookup algorithm.

6.7.4.2 Reference values

Note that the size of FIB depends on the domain's degree (the number of egress links), so the evaluation should be performed for the largest Internet domains (tire 1). The FIB size of CAFE should be reduced comparing to the size of FIB used in IP routers.

6.7.5 Size of COMET header

6.7.5.1 Definition

The COMET header size defines the number of additional bytes (comet header) that must be placed in data packets to use COMET forwarding technique. Note that the size of COMET header depends on the length of content delivery paths, so the evaluation should be performed for the Internet scale network.

6.7.5.2 Reference values

The overhead introduced by COMET should not exceed the order of IPv6 header (40B) for the Internet scale network.

6.7.6 Hop count

6.7.6.1 Definition

The hop count defines the number of AS-level hops required for a particular content to reach the client from the server. In the coupled approach, a comparison will be made between the number of hops along the original resolution path, and that along the route-optimised path. This metrics will also be used to compare both COMET approaches.

6.7.6.2 Reference values

The average hop count of the content delivery path after route optimisation should not exceed 7–8 hops, as reported independently by Wang *et al* in [30], and Cho *et al* in [31]. Similar values for the number of caching entities along the path have been used in our caching studies in [32] and [33]. In these studies we show that the number of hops to be traversed can also be reduced if efficient caching algorithms are in place.

6.7.7 Bandwidth consumption

6.7.7.1 Definition

The bandwidth consumption defines the bandwidth saving achieved under multicast over unicast. This is represented in relation to the number of hops. Therefore, if the aggregate hop count for a given content delivery under unicast is H_u, and the aggregate under multicast is H_m, then the bandwidth saving will be given by $(H_u - H_m)/H_u$. Bandwidth savings have also been reported in [32], in terms of the cache hits. We have shown that efficient caching can save up to one order of magnitude in the bandwidth requirement.

6.7.7.2 Reference values

This is a comparison between two schemes, so the bandwidth consumption of unicast is essentially the reference, and will be determined by experimentation.

7 Summary and conclusions

In this deliverable, we reported activities related to software integration, validation of developed prototypes, and integration of applications. Following the COMET architecture, we focused on two complementary prototypes related to the COMET coupled and decoupled approaches. Taking into account the differences in both approaches for each of them, we defined specific integration environment, the integration testbed build on virtualisation platforms, and a number of validation tests. Validation of both prototypes followed the bottom-up approach, where we began from the validation of particular entities or sub-processes and we finished with the overall scenarios proving proper inter-working of entities and overall processes. The performed validation tests allowed detecting and fixing bugs in the developed software, discovering and fixing open issues in the system specification as well as identifying some bottlenecks in the developed software.

The integration of applications with the COMET system focused on 4 types of applications that are: Video streaming, Video on Demand, Peer-to-peer application, and Content Streaming Relay (CSR) supporting point-to-multipoint connections. These applications were selected to demonstrate distribution of live and pre-recorded content in point-to-point, peer-to-peer and point-to-multipoint scenarios.

Following reviewers' recommendations, we identified a set of performance metrics. These metrics correspond to both the overall COMET system performance as well as the benchmarks related to major COMET processes. In particular, the proposed metrics focus on: (1) overall content resolution and retrieval process, (2) content publication process, (3) name resolution process, (4) server and network awareness process, (5) routing awareness process, and (6) content forwarding process. For each metric, we proposed the approach to assess obtained results and, whenever it was possible, we identified the reference values based on analysis of related works on similar systems.

The key conclusions are:

1. The performed integration and validation tests related to both prototypes confirm that:
    - both approaches are feasible for implementation,
    - the functions realized by developed entities and the interfaces between them are conformable to the specification,
    - the main COMET processes related to content publication, resolution and delivery are properly carried out allowing for proper content consumption.
2. The incremental development approach with three development cycles including specification, implementation, integration and validation allowed efficient code development by verifying specification of main components at early stage, providing feedback to the designers about identified open issues, focusing developers, integrators and tester on specific tasks.
3. The proposed integration and validation framework with associated integration tools, i.e. svn, track, hudson, effectively support integration process by keeping track of code changes, simplifying communication between developers, integrators and testers.
4. The validation testbed build based on virtual environment significantly simplifies validation process because it allowed relatively easy repeat tests' conditions. Moreover, we were able to validate COMET software in relatively large network environment consisting of 32 nodes.
5. The bottom-up validation approach allowed identifying bugs and open issues in the developed software and inconsistencies in specification.
6. The lessons from adaptation of 4 applications point out that the COMET system:
    - may be deployed without significant modifications of the consumer terminals or content servers. The required adaptation are: installation of content client module which provide interface to the COMET system, and server monitoring module to provide information about the status of content server,
    - is flexible to interwork with other content distribution services as per-to-peer or point-to-multipoint.

8 References

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

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

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

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

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

[6] http://staruml.sourceforge.net/en

[7] http://www.java.com/en

[8] http://ant.apache.org

[9] http://www.mingw.org

[10] http://code.google.com/p/waf

[11] http://subversion.tigris.org

[12] http://www.spirent.com

[13] COMET Deliverable, "D6.1: Demonstration Scenarios and Test Plan", June 2012.

[14] http://www.videolan.org/vlc

[15] http://httpd.apache.org

[16] http://www.utorrent.com/intl/es

[17] http://www.tornadoweb.org

[18] Jarno Rajahalme, Mikko Särelä, Kari Visala, and Janne Riihijärvi, "On name-based inter-domain routing," Computer and Telecommunications Networking, Computer Networks: The International Journal of, vol. 55, no. 4, March 2011, pp. 975–986.

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

[20] Fiona Fui-Hoon Nah, "A study on tolerable waiting time: how long are Web users willing to wait?", Behaviour & Information Technology, vol. 23 issue 3, 2004

[21] http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/

[22] Akamai, "Boosting Online Commerce Profitability with Akamai," Akamai Technologies, 2007, http://www.akamai.com

[23] Recommendations for web page developers http://www.technologyreview.com/files/54902/GoogleSpeed_charts.pdf

[24] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris, "DNS Performance and the Effectiveness of Caching", IEEE/ACM Transactions on Networking (TON) archive, Volume 10 Issue 5, October 2002 Pages 589 – 603

[25] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian, "Delayed Internet routing convergence", SIGCOMM Comput. Commun. Rev. 30, 4, August 2000, pp. 175-187.

[26] The Cooperative Association for Internet Data Analysis, http://www.caida.org/.

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

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

[29] C. Popoviciu, A. Hamza, G. Van de Velde, D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", IETF document RFC5180, May 2008, http://www.ietf.org/rfc/rfc5180.txt

[30] Yi Wang, Shaozhi Ye, and Xing Li, "Understanding current IPv6 performance: a measurement study," *in Proc.* 10th IEEE Symposium on Computers and Communications (ISCC'05), June 2005, pp. 71–76.

[31] Kideok Cho, Munyoung Lee, Kunwoo Park, Ted Taekyoung Kwon, Yanghee Choi, and Sangheon Pack, "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," *in Proc.* IEEE INFOCOM Workshops, March 2012, pp. 316-321.

[32] Ioannis Psaras, Wei Koong Chai, George Pavlou, "Probabilistic In-Network Caching for Information-Centric Networks", ACM Sigcomm Workshop on ICN, August 2012.

[33] Wei Koong Chai, Diliang He, Ioannis Psaras, George Pavlou, Cache "Less for More" in Information-Centric Networks, IFIP Networking 2012.

9 Abbreviations

AS	Autonomous System
ATE	Automatic Test Equipment
BGP	Border Gateway Protocol
BW	Bandwidth
CAFE	Content Aware Forwarding Entity
CC	Content Client
CME	Content Mediation Entity
CoS	Class Of Service
CP	Content Publisher
CR	Content Record
CRE	Content Resolution Entity
CRL	Content Retrieval Latency
CRME	Content Resolution and Mediation Entity
CRSR	Content Retrieval Success Ratio
CRT	Content Resolution Time
CS	Content Server
CSR	Content Streaming Relay
CSS	Content Streaming Server
DB	Database
DNS	Domain Name System
DUT	Device Under Test
FIB	Forwarding Information Base
HS	Handle System
HTTP	HyperText Transfer Language
IP	Internet Protocol
IPLR	IP Packet Loss Ratio
IPTD	IP Packet Transfer Delay
IPv4	IP version 4
IPv6	IP version 6
ISP	Internet Service Provider
JVM	Java Virtual Machine
MPEG	Moving Picture Expert Group
NLRI	Network Layer Reachability Information
QoS	Quality of Service
RAE	Routing Awareness Entity
RCT	Routing Convergence Time

RTSP	Real-Time Streaming Protocol
RTT	Round Trip Time
SIC	Server Information Collector
SMA	Server Monitoring Agent
SNME	Server and Network Monitoring Entity
SW	Software
TCP	Transport Class Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Universal Resource Locator
VLC	VideoLan Player
VM	Virtual Machine
VoD	Video On Demand

10 Appendix A: Integration and validation of release 1

10.1 Test Results: Publication (CP – CRE)

Connecting to the Content Publisher which runs as a web server and can be accessed via the URL: http://10.5.90.245:8080/Publisher/ (IP local to the testbed)

Test Category	Test	Test Description	Step	Expected	Actual	Result
1. Publication	1	Incorrect credentials 1	Navigate to http://10.5.90.245:8080/Publisher	Error Message: "Wrong Credentials!"	Error Message: "Wrong Credentials!"	PASS
			Enter incorrect credentials: [Admin Handle: COMET/ADMIN] , [Password: comet]			
			Click "Login" button			
	2	Incorrect credentials 2	Navigate to http://10.5.90.245:8080/Publisher	Error Message: "Correct format of admin handle is .../ADMIN!"	Error Message: "Correct format of admin handle is .../ADMIN!"	PASS
			Enter incorrect credentials: [Admin Handle: com.gmail@geopet/ADMIN] , [Password: geopet]			
			Click "Login" button			
	3	Correct credentials	Navigate to http://10.5.90.245:8080/Publisher	Successful login & Redirected to: http://10.5.90.245:8080/Publisher/LoginServlet (Administration Page)	Successful login & Redirected to: http://10.5.90.245:8080/Publisher/LoginServlet (Administration Page)	PASS
			Enter incorrect credentials: [Admin Handle: COMET/ADMIN] , [Password: password]			
			Click "Login" button			

2. Content Registration	4	Registration of "COMET/my_program"	Follow Test 3 to successfully authenticate	Receive Success message "Success! Handle COMET/my_program was created with CID [some_cid]"	Receive Success message "Success! Handle COMET/my_program was created with CID [some_cid]"	**PASS**
			From the Administratoin page select "Create a New Handle" and click "Go" button			
			Fill in the form with the following information: [Content Name: COMET/my_program] , [Content Type: video/x-flv], [CoS: BE], [Number of QoS constraints: 1], [QoS constraint #1: 0x070032], [Number of Traffic Descriptors: 1], [Traffic Descriptor #1: 0x080190], [priority: 5], [Application protocol: http], [Transport protocol: tcp], [Transport port: 80], [Number of Content Servers: 2], [Content Server #1 - IP address: 10.5.10.138], [Content Server #1 - Path: /test.flv], [Content Server #1 - Load: 1], [Content Server #1 - CME ID: cme://1.1.1.1], [Content Server #2 - IP address: 10.5.10.185:5050], [Content Server #2 - Path: /test2.flv], [Content Server #2 - Load: 1], [Content Server #2 - CME ID: cme://#12345]			
			Click "Add New Source" button			
			Fill in the form with the following information: [Content Type: video/x-flv], [CoS: BE], [Number of QoS constraints: 2], [QoS constraint #1: 0x0903E8], [QoS constraint #1: 0x070028], [Number of Traffic Descriptors: 1], [Traffic Descriptor #1: 0x080190], [priority: 10], [Application protocol: rtsp], [Transport protocol: udp], [Transport port: 1417], [Number of Content Servers: 1], [Content Server #1 - IP address: 10.5.10.138], [Content Server #1 - Path: /test3.flv], [Content Server #1 - Load: 1], [Content Server #1 - CME ID: cme://1.1.1.1]			
			Click "Create" button			

	5	Registration of "COMET/birthday_2010"	Follow Test 3 to successfully authenticate From the Administration page select "Create a New Handle" and click "Go" button Fill in the form with the following information: [Content Name: COMET/birthday_2010] , [Content Type: video/mp4], [CoS: BTBE], [Number of QoS constraints: 1], [QoS constraint #1: 0x070032], [Number of Traffic Descriptors: 1], [Traffic Descriptor #1: 0x080190], [priority: 5], [Application protocol: rtsp], [Transport protocol: udp], [Transport port: 675], [Number of Content Servers: 1], [Content Server #1 - IP address: 10.5.10.138], [Content Server #1 - Path: /test.flv], [Content Server #1 - Load: 1], [Content Server #1 - CME ID: cme://1.1.1.1] Click "Create" button	Receive Success message "Success! Handle COMET/my_program was created with CID [some_cid]"	Receive Success message "Success! Handle COMET/my_program was created with CID [some_cid]"	**PASS**
3. Content Records Update	6	Content source modification	Follow Test 3 to successfully authenticate. Assuming Tests 4 and 5 already registered content we proceed. From the Administration page select "Edit an Existing Handle" and click "Go" button Fill in the box "Choose the handle you want to modify::" with "COMET/my_program", select "Edit/Delete a Content Source" and click "Go" button" Select 1st Source from "Select the Content Source you want to edit:" dropdown and click "Edit" button Raise "priority" to 7 from 5 Click "Update" button	Receive Success message "Success! Content Source at index 2 was edited!"	Receive Success message "Success! Content Source at index 2 was edited!"	**PASS**

	7	Content source addition	Follow Test 3 to successfully authenticate. Assuming Tests 4 and 5 already registered content we proceed. From the Administration page select "Edit an Existing Handle" and click "Go" button Fill in the box "Choose the handle you want to modify::" with "COMET/birthday_2010", select "Add a new Content Source" and click "Go" button" Fill in the form with the following information: [Content Type: video/mp4], [CoS: PR], [Number of QoS constraints: 1], [QoS constraint #1: 0x0903E8], [QoS constraint #2: 0x070028], [Number of Traffic Descriptors: 1], [Traffic Descriptor #1: 0x080190], [priority: 10], [Application protocol: http], [Transport protocol: tcp], [Transport port: 80], [Number of Content Servers: 2], [Content Server #1 - IP address: 10.5.10.138], [Content Server #1 - Path: /test.flv], [Content Server #1 - Load: 1], [Content Server #1 - CME ID: cme://1.1.1.1], [Content Server #2 - IP address: 10.5.10.185:5050], [Content Server #2 - Path: /test2.flv], [Content Server #2 - Load: 1], [Content Server #2 - CME ID: cme://#12345] Click "Add" button	Receive Success message "Success! Content Source at index 3 was edited!"	Receive Success message "Success! Content Source at index 3 was edited!"	PASS
	8	Content server modification	Follow Test 3 to successfully authenticate. Assuming Tests 4 and 5 already registered content we proceed. From the Administration page select "Edit an Existing Handle" and click "Go" button Fill in the box "Choose the handle you want to modify::" with "COMET/birthday_2010", select "Edit/Delete a Content Source" and click "Go" button"	Receive Success message "Success! Content Source at index 2 was edited!"	Receive Success message "Success! Content Source at index 2 was edited!"	PASS

			Select 1st Source from "Select the Content Source you want to edit:" dropdown and click "Edit" button			
			Change IP address of Content Server #1 to 4.3.2.1			
			Click "Update" button			
	9	Content server deletion	Follow Test 3 to successfully authenticate. Assuming Tests 4 and 5 already registered content we proceed.	Receive Success message "Success! Content Source at index 2 was edited!"	Receive Success message "Success! Content Source at index 2 was edited!"	PASS
			From the Administration page select "Edit an Existing Handle" and click "Go" button			
			Fill in the box "Choose the handle you want to modify::" with "COMET/my_program", select "Edit/Delete a Content Source" and click "Go" button"			
			Select 1st Source from "Select the Content Source you want to edit:" dropdown and click "Edit" button			
			Select Delete next to "Content Server #2:"			
			Click "Update" button			
	10	Content server addition	Follow Test 3 to successfully authenticate. Assuming Tests 4 and 5 already registered content we proceed. And that you have deleted the second content server through Test 9	Receive Success message "Success! Content Source at index 2 was edited!"	Receive Success message "Success! Content Source at index 2 was edited!"	PASS
			From the Administration page select "Edit an Existing Handle" and click "Go" button			
			Fill in the box "Choose the handle you want to modify::" with "COMET/my_program", select "Edit/Delete a Content Source" and click "Go" button"			
			Select 1st Source from "Select the Content Source you want to edit:" dropdown and click "Edit" button			

			Fill in the box "Add more Content Servers:" with "1"			
			Fill in the form with the following information: [Content Server #2 - IP address: 2001:0DB8:AC10:FE10::], [Content Server #2 - Path: /comet/images/comet_demo.flv], [Content Server #2 - Load: 1], [Content Server #2 - CME ID: cme://#12345]			
			Click "Update" button			
	11	Verify that new content server was added to handle COMET/my_program	Assuming we have run everything up to Test 10	Information for Content Server #2 is: [Content Server #2 - IP address: 2001:0DB8:AC10:FE10::], [Content Server #2 - Path: /comet/images/comet_demo.flv], [Content Server #2 - Load: 1], [Content Server #2 - CME ID: cme://#12345]	Information for Content Server #2 is: [Content Server #2 - IP address: 2001:0DB8:AC10:FE10::], [Content Server #2 - Path: /comet/images/comet_demo.flv], [Content Server #2 - Load: 1], [Content Server #2 - CME ID: cme://#12345]	**PASS**
			From the Administration page select "Edit an Existing Handle" and click "Go" button			
			Fill in the box "Choose the handle you want to modify::" with "COMET/my_program", select "Edit/Delete a Content Source" and click "Go" button"			
			Select 1st Source from "Select the Content Source you want to edit:" dropdown and click "Edit" button			
			Check that Content Server #2 contains correct information			

10.2 Test Results: Content Resolution (CC-CME-CRE)

For the second test case described in Section requires the use the Client.exe file which communicates with the CME to consume content

Test Category	Test	Test Description	Step	Expected	Actual	Result	Comments
Contetn resolution	1	Request content existing from CME	1. Make sure CME server is running. CME ip is 10.5.90.244	Retrieve content from CME server. In output.dat file must be " CONTENT CLIENT: RECEIVED DATA" section	########################## ########## # CONTENT CLIENT: RECEIVED DATA # ########################## ########## [HEADER] ID: 06 QRITZCODE: 0xC000 QDCOUNT: 0x0001 ANCOUNT: 0x0001 NSCOUNT: 0x0001 ASCOUNT: 0x0001 [QUERY] QNAME: COMET/BIRTHDAY_2010_10235 QTYPE: 0x0064 QCLASS: 0x0001 [ANSWER] CID: 1e9b8ec89bf48c6bc2368a666ce4a531 MIME TYPE: video/mp4 APP/SESSION PROTOCOL: 14 TRANSPORT PROTOCOL: 17 SERVER PORT: 675 SERVER IP: 1.1.1.2 PATH SIZE: 37	PASS	Client.exe receives path corresponding to COMET/BIRTHDAY_2010_10235, and stores it in output.dat
			2. Login into CME web interface and choose 'configure CRE' and provide ip of CRE and port to what CRE is listening. CRE ip is 10.5.90.245. CRE port is 2641				No browser launched with found path
			3. Launch Client.exe and provide 1. remote IP adderss of cme 10.5.90.244 2. remote port: 9091. Existing content name. COMET/BIRTHDAY_2010_10235				

					PATH: /user/profiles/john_doe/birthday_2010		
2. Client requests from CME non-existing content	**2**	Client requests from CME non-existing content (failure testing).	1. Make sure CME server is running. CME ip is 10.5.90.244 2. Login into CME web interface and choose 'configure CRE' and provide ip of CRE and port to what CRE is listening. CRE ip is 10.5.90.245. CRE port is 2641 3. Launch Client.exe and provide 1. remote IP adderss of cme 10.5.90.244 2. remote port: 9091. Existing content name. COMET/XYZ	Message saying that content name cannot be solved	In cme.log found message: WARN DNSHandler - Received Content Name cannot be resolved to Content Record!	**PASS**	On client.exe side there are no infomative message saying that content cannot be solved. It would be nice to have one.

10.3 Test Results: Routing awareness (CME – RAE)

For the third test case described requires both CME and the RAE running and validation is done through examining the logs and seeing if interaction between them takes place and if appropriate RAE messages were recognized and executed by the CME.

Test Category	Test	Test Description	Step	Expected	Actual	Result
1. RAE updates information about paths	1	Add prefix	Configure RAE and startup 7 instances. Check cme.log	In logs "Message of type INSERT_PATHS received @ RAE interface from SOME_IP"	In logs "Message of type INSERT_PATHS received @ RAE interface from SOME_IP"	PASS
	2	Remove prefix	Configure RAE and startup 7 instances. Check cme.log	In logs "Message of type REMOVE_PATHS received @ RAE interface from SOME_IP"	In logs "Message of type REMOVE_PATHS received @ RAE interface from SOME_IP"	PASS
	3	Update information about paths	Configure RAE and startup 7 instances. Check cme.log	In logs "Message of type INSERT_PATHS received @ RAE interface from SOME_IP"	In logs "Message of type INSERT_PATHS received @ RAE interface from SOME_IP"	PASS

2. RAE updates information about domain provisioning	**4**	Update information about domain provisioning	Configure RAE and startup 7 instances. Check cme.log	In logs "Message of type INSERT_PROVISIONING received @ RAE interface from SOME_IP"	In logs "Message of type INSERT_PROVISIONING received @ RAE interface from SOME_IP"	**PASS**
3. Reset/unavailability of RAE	**5**	Reset/unavailablelitiy of RAE	Configure RAE and startup 7 instances. Check cme.log	In logs "Message of type RESET received @ RAE interface from SOME_IP"	In logs "Message of type RESET received @ RAE interface from SOME_IP"	**PASS**
4. Stress tests	**6**	Updating large number of prefixes	Setup topology: CME at 10.5.90.244:9090 ^ |	Updated large number of prefixes	Updated large number of prefixes	**PASS**

			DOMAIN 1 <-------link-------> DOMAIN2 | | + +++++ single prefix COUNT prefixes 1.0.0.0/8 2.x.y.z/32			

11 Appendix B: Integration and validation of release 2

11.1 Test Results: Publication (CP – CRE)

Test Category	Test #	Test	Expected	Actual	Result
Content registration	**1**	Publication in domain 1 of content available in streaming and download	Records created and accessible from CP administration page	Records created and accessible from CP administration page	**PASS**
	2	Publication in domain 2 of content available in streaming and download	Records created and accessible from CP administration page	Records created and accessible from CP administration page	**PASS**
	3	Publication of content with false/unreal data	Record cannot be created and receive error message	Record cannot be created and receive error message	**PASS**
Content update/edit	**3**	Alter/edit content created in test 1	Records updated and alteration visible from CP administration page	Records updated and alteration visible from CP administration page	**PASS**
	4	Alter/edit content created in test 2	Records updated and alteration visible from CP administration page	Records updated and alteration visible from CP administration page	**PASS**

Delete content	**5**	Delete content created in all domains	No records are available or present	No records are available or present	**PASS**
Batch update/edit and delete	**6**	Batch publication in domain 1 and 2	Records created and accessible from CP administration page	Records created and accessible from CP administration page	**PASS**
	7	Batch edit in domain 1 and 2	Records edited and updated in CP administration page	Records edited and updated in CP administration page	**PASS**
	8	Batch delete in domain 1 and 2	Records deleted and not available in CP administration page	Records edited and updated in CP administration page	**PASS**
	9	Batch edit, update and delete in domain 1 and 2	Records created, updated and deleted. All set of actions were executed.	Records created, updated and deleted. All set of actions were executed.	**PASS**
	10	Batch publications, edit, and delete using false commands, non-existing content and false/unreal data.	Batch execution not performed and receive error message	Batch execution not performed and receive error message	**PASS**

11.2 Test Results: Decision process (exclude CAFE-Server awareness)

Test Category	Test #	Test	Expected	Actual	Result
CoS	11	Consume content in domain 1 with CoS BE using a CC with CoS BE or BTBE or PR in domain 2 or 3	Content delivered without generating paths	Content delivered without generating paths	PASS
	12	Consume content in domain 1 with CoS BTBE with available paths in PR and BTBE using a CC with CoS BE or BTBE in domain 2 or 3	Content delivered via BTBE paths	Content delivered via BTBE paths	PASS
	13	Consume content in domain 1 with CoS PR with available paths in PR and BTBE using a CC with CoS BTBE or BE in domain 2 or 3	No paths found and effectively no content delivery	No source (CS) found and effectively no content delivery	PASS
	14	Consume content in domain 2 with CoS BE using a CC with CoS BE or BTBE or PR in domain 1 or 3	Content delivered without generating paths	Content delivered without generating paths	PASS
	15	Consume content in domain 2 with CoS BTBE with available paths in PR and BTBE using a CC with CoS BTBE or PR in domain 1 or 3	Content delivered via BTBE paths	Content delivered via BTBE paths	PASS

	16	Consume content in domain 2 with CoS PR with available paths in PR and BTBE using a CC with CoS BTBE or BE in domain 1 or 3	No paths found and effectively no content delivery	No source (CS) found and effectively no content delivery	**PASS**
Path length	**17**	Consume content restricted to a path length of 3 from CS domain 1 using a CC from domain 2 or 3	Two paths found and the best path decided is the shortest. Content delivered	Two paths found and the best path decided is the shortest. Content delivered	**PASS**
	18	Consume content restricted to a path length of 2 from CS domain 1 using a CC from domain 2 or 3	One path found. Content delivered	One path found. Content delivered	**PASS**
	19	Consume content restricted to a path length of 3 from CS domain 2 using a CC from domain 1 or 3	Two paths found and the best path decided is the shortest. Content delivered	Two paths found and the best path decided is the shortest. Content delivered	**PASS**
	20	Consume content restricted to a path length of 2 from CS domain 2 using a CC from domain 1 or 3	Two paths found and the best path decided is the shortest. Content delivered	Two paths found and the best path decided is the shortest. Content delivered	**PASS**
IPTD, IPLR	**21**	Consume content requiring IPTD set at minimum delay recorded from paths available between domain 1 and (2 or 3)	One path selected i.e. the one with smallest delay. Content delivered	One path selected i.e. the one with smallest delay. Content delivered	**PASS**
	22	Consume content requiring IPTD set at maximum delay recorded from paths available between domain 1 and (2 or 3)	Two paths available, but the one with smallest delay is selected. Content delivered	Two paths available, but the one with smallest delay is selected. Content delivered	**PASS**

	23	Consume content requiring IPTD set at minimum delay recorded from paths available between domain 2 and (1 or 3)	One path selected i.e. the one with smallest delay. Content delivered	One path selected i.e. the one with smallest delay. Content delivered	**PASS**
	24	Consume content requiring IPTD set at maximum delay recorded from paths available between domain 2 and (1 or 3)	Two paths available, but the one with smallest delay is selected. Content delivered	Two paths available, but the one with smallest delay is selected. Content delivered	**PASS**
	25	Consume content requiring IPLR set at minimum loss recorded from paths available between domain 1 and (2 or 3)	One path selected i.e. the one with smallest loss. Content delivered	One path selected i.e. the one with smallest loss. Content delivered	**PASS**
	26	Consume content requiring IPLR set at maximum loss recorded from paths available between domain 1 and (2 or 3)	Two paths available, but the one with smallest loss is selected. Content delivered	Two paths available, but the one with smallest loss is selected. Content delivered	**PASS**
	27	Consume content requiring IPLR set at minimum loss recorded from paths available between domain 2 and (1 or 3)	One path selected i.e. the one with smallest loss. Content delivered	One path selected i.e. the one with smallest loss. Content delivered	**PASS**
	28	Consume content requiring IPLR set at maximum loss recorded from paths available between domain 2 and (1 or 3)	Two paths available, but the one with smallest loss is selected. Content delivered	Two paths available, but the one with smallest loss is selected. Content delivered	**PASS**
Server load	**29**	Consume content when server load type tolerant	The server still selected even when disconnected	The server still selected even when disconnected	**PASS**

	30	Consume content when server load type strict	The server excluded and not selected	The server excluded and not selected	PASS
	31	Consume content available from two servers with one server with status L and the other status H	Server with status L selected i.e. with highest rank	Server with status L selected i.e. with highest rank	PASS

11.3 Test Results: Server awareness (SNME-SMA)

Test Category	Test #	Test	Expected	Actual	Result
SNME-SMA	**1**	CS 10.2.0.3 is at level L	CS 10.2.0.3 status in SNME is L	CS 10.2.0.3 status in SNME is L	PASS
	2	CS 10.2.0.3 is at level M	CS 10.2.0.3 status in SNME is M	CS 10.2.0.3 status in SNME is M	PASS
	3	CS 10.2.0.3 is at level H	CS 10.2.0.3 status in SNME is H	CS 10.2.0.3 status in SNME is H	PASS

	4	CS 10.2.0.3 is at level L based on new levels	CS 10.2.0.3 status in SNME is L	CS 10.2.0.3 status in SNME is L	PASS
	5	CS 10.2.0.3 is at level M based on new levels	CS 10.2.0.3 status in SNME is M	CS 10.2.0.3 status in SNME is M	PASS
	6	CS 10.2.0.3 is at level H based on new levels	CS 10.2.0.3 status in SNME is H	CS 10.2.0.3 status in SNME is H	PASS
	7	CS 10.2.0.3 is disconnected from network	CS 10.2.0.3 status in SNME is U and then D	CS 10.2.0.3 status in SNME is U and then D	PASS
	8	CS 10.2.0.3 is disconnected from network and then connected again.	CS 10.2.0.3 status in SNME is U or D and then back to the appropriate level L or M or H (i.e. not dead)	CS 10.2.0.3 status in SNME is U or D and then back to the appropriate level L or M or H (i.e. not dead)	PASS
	9	CS 10.1.0.3, 10.1.0.13, 10.2.0.3 and 10.2.0.13 are at different status	Correct updated status for each CS is found in SNME	Correct updated status for each CS is found in SNME	PASS
	10	Disconnect one or more CS	Correct updated status for each CS is found in SNME	Correct updated status only for live CSes, but not for dead CSes. Status idle for dead CSes.	FAIL

11.4 Test Results: COMET system (exclude CAFE-Server awareness)

Test Category	Test #	Test	Expected	Actual	Result
Intra-Domain	1	Publish static content in domain 1 and request from local CC	Content delivered to CC	Content delivered to CC	PASS
	2	Publish static content in domain 1 and request from local CC	Content delivered to CC	Content delivered to CC	PASS
Inter-Domain	3	Publish static content in domain 1 and request from CC domain 2	Content delivered to CC	Content delivered to CC	PASS
	4	Publish static content in domain 1 and request from CC domain 3	Content delivered to CC	Content delivered to CC	PASS
	5	Publish static content in domain 2 and request from CC domain 1	Content delivered to CC	Content delivered to CC	PASS
	6	Publish static content in domain 2 and request from CC domain 3	Content delivered to CC	Content delivered to CC	PASS

Non-static content	**7**	Publish non-static content (e.g. rtsp) in domain 1 and request from CC domain 2	Content delivered to CC	Content delivered to CC	**PASS**
	8	Publish non-static content in domain 1 and request from CC domain 3	Content delivered to CC	Content delivered to CC	**PASS**
	9	Publish non-static content in domain 2 and request from CC domain 1	Content delivered to CC	Content delivered to CC	**PASS**
	10	Publish non-static content in domain 2 and request from CC domain 3	Content delivered to CC	Content delivered to CC	**PASS**

12 Appendix C: Integration and validation of release 3

12.1 Test results: Path configuration

Test Category	Test #	Test	Expected	Actual	Result
CME path configuration	1	Consumption between access network 111.111.111.0/24 CS and 101.101.101.0/24 CC Path generated by CME, configured to agent CAFE and equivalent path followed	Generated and configured path: e2 Packet flow corresponding to path: CAFEedge2 (domain 1) -> CAFEedge1 (domain 1).	Generated and configured path: e2 Packet flow corresponding to path: CAFEedge2 (domain 1) -> CAFEedge1 (domain 1).	PASS
	2	Consumption between access network 101.101.101.0/24 CS and 111.111.111.0/24 CC Path generated by CME, configured to agent CAFE and equivalent path followed	Generated and configured path: e1 Packet flow corresponding to path: CAFEedge1 (domain 1) -> CAFEedge2 (domain 1).	Generated and configured path: e1 Packet flow corresponding to path: CAFEedge1 (domain 1) -> CAFEedge2 (domain 1).	PASS
	3	Consumption between access network 101.101.101.0/24 CS and 222.222.222.0/24 CC Path generated by CME, configured to agent CAFE and equivalent path followed	Generated and configured path: 22c1b211 CAFEedge1 (domain1) -> CAFE forward (domain 1) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 2) -> CAFEedge2 (domain2).	Generated and configured path: 22c1b211 CAFEedge1 (domain1) -> CAFE forward (domain 1) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 2) -> CAFEedge2 (domain2).	PASS
	4	Consumption between access network 111.111.111.0/24 CS and 202.202.202.0/24 CC Path generated by CME, configured to agent CAFE and equivalent path followed	Generated and configured path: 22c1b201 CAFEedge2 (domain1) -> CAFE forward (domain 1) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 2) -> CAFEedge1 (domain2).	Generated and configured path: 32c1b201 CAFEedge2 (domain1) -> CAFE forward (domain 1) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 2) -> CAFEedge1 (domain2).	PASS

	5	Consumption between access network 222.222.222.0/24 CS and 111.111.111.0/24 CC Path generated by CME, configured to agent CAFE and equivalent path followed	Generated and configured path: 12b1c231 Packet flow: CAFEedge2 (domain2) -> CAFE forward (domain 2) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 1) -> CAFEedge2 (domain1).	Generated and configured path: 12b1c231 Packet flow: CAFEedge2 (domain2) -> CAFE forward (domain 2) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 1) -> CAFEedge2 (domain1).	**PASS**
	6	Consumption between access network 111.111.111.0/24 CS and 101.101.101.0/24 CC Agent CAFE disconnected	Consumption terminates at path configuration stage when agent not responding	Consumption terminates at path configuration stage when agent not responding	**PASS**

12.2 Test results: Decision Process (BW management)

Test Category	Test #	Test	Expected	Actual	Result
Inter Domain CME BW management	**1**	Multiple consumptions from PR CC of same content available in PR and BTBE paths and sources Between domain 1 (CS) and domain 2 (CC)	Consumption and paths: 1. 202.202.202.2 PR CC : 1,3,2 PR 2. 222.222.222.2 PR CC : 1,3,2 PR 3. 222.222.222.22 PR CC : 1,2 PR (agr. exceed) 4. 202.202.202.2 PR CC: 1,2 PR 5. 222.222.222.2 PR CC: 1,3,2 BTBE (agr. exceed and 2nd BTBE select)	Consumption and paths: 1. 202.202.202.2 PR CC : 1,3,2 PR 2. 222.222.222.2 PR CC : 1,3,2 PR 3. 222.222.222.22 PR CC : 1,2 PR (agr. exceed) 4. 202.202.202.2 PR CC: 1,2 PR 5. 222.222.222.2 PR CC: 1,3,2 BTBE (agr. exceed and 2nd BTBE select)	**PASS**
	2	Multiple consumptions from PR CC of same content available in PR and BTBE paths and sources Between domain 2 (CS) and domain 1 (CC)	Consumption and paths: 1. 101.101.101.2 PR CC : 2,3,1 PR 2. 111.111.111.2 PR CC : 2,3,1 PR 3. 111.111.111.22 PR CC : 2,1 PR (agr. exceed) 4. 101.101.101.2 PR CC: 2,1 PR 5. 111.111.111.2 PR CC: 2,3,1 BTBE (agr. exceed and 2nd BTBE select)	Consumption and paths: 1. 101.101.101.2 PR CC : 2,3,1 PR 2. 111.111.111.2 PR CC : 2,3,1 PR 3. 111.111.111.22 PR CC : 2,1 PR (agr. exceed) 4. 101.101.101.2 PR CC: 2,1 PR 5. 111.111.111.2 PR CC: 2,3,1 BTBE (agr. exceed and 2nd BTBE select)	**PASS**

	3	Multiple consumptions of BTBE CC same content available in PR and BTBE paths and sources Between domain 1 (CS) and domain 2 (CC)	Consumption and paths: 1. 202.202.202.2 BTBE CC : 1,3,2 BTBE 2. 222.222.222.2 BTBE CC : 1,3,2 BTBE 3. 222.222.222.22 BTBE CC : 1,3,2 BTBE 4. 222.222.222.2 BTBE CC : 1,3,2 BTBE 5. 222.222.222.22 BTBE CC : 1,3,2 BTBE 6. 222.222.222.22 BTBE CC : 1,3,2 BTBE BTBE consumptions should be mediated but no reservations are performed to link i.e. BTBE along a path are unlimited	Consumption and paths: 1. 202.202.202.2 BTBE CC : 1,3,2 BTBE 2. 222.222.222.2 BTBE CC : 1,3,2 BTBE 3. 222.222.222.22 BTBE CC : 1,3,2 BTBE 4. 222.222.222.2 BTBE CC : 1,3,2 BTBE 5. 222.222.222.22 BTBE CC : 1,3,2 BTBE 6. 222.222.222.22 BTBE CC : 1,3,2 BTBE BTBE consumptions should be mediated but no reservations are performed to link i.e. BTBE along a path are unlimited	**PASS**
	4	Multiple consumptions of BTBE CC same content available in PR and BTBE paths and sources Between domain 2 (CS) and domain 1 (CC)	Consumption and paths: 1. 101.101.101.1 BTBE CC : 2,3,1 BTBE 2. 111.111.111.2 BTBE CC : 2,3,1 BTBE 3. 111.111.111.22 BTBE CC : 2,3,1 BTBE 4. 101.101.101.1 BTBE CC : 2,3,1 BTBE 5. 111.111.111.2 BTBE CC : 2,3,1 BTBE 6. 111.111.111.22 BTBE CC : 2,3,1 BTBE BTBE consumptions should be mediated but no reservations are performed to link i.e. BTBE along a path are unlimited	Consumption and paths: 1. 101.101.101.1 BTBE CC : 2,3,1 BTBE 2. 111.111.111.2 BTBE CC : 2,3,1 BTBE 3. 111.111.111.22 BTBE CC : 2,3,1 BTBE 4. 101.101.101.1 BTBE CC : 2,3,1 BTBE 5. 111.111.111.2 BTBE CC : 2,3,1 BTBE 6. 111.111.111.22 BTBE CC : 2,3,1 BTBE BTBE consumptions should be mediated but no reservations are performed to link i.e. BTBE along a path are unlimited	**PASS**

12.3 Test results: Content Forwarding (Intra/Inter Domain)

Test Category	Test #	Test	Expected	Actual	Result
Intra Domain Content Forwarding	1	CAFE interception when intra domain consumption between access network 111.111.111.0/24 CS and 101.101.101.0/24 CC	Packets intercepted by CAFEedge2 and forwarded to CAFEedge1 in domain 1. Consumption of video content is unaffected.	Packets intercepted by CAFEedge2 and forwarded to CAFEedge1 in domain 1. Consumption of video content is unaffected.	PASS
	2	CAFE interception when intra domain consumption between access network 101.101.101.0/24 CS and 111.111.111.0/24 CC	Packets intercepted by CAFEedge1 and forwarded to CAFEedge2 in domain 1. Consumption of video content is unaffected.	Packets intercepted by CAFEedge1 and forwarded to CAFEedge2 in domain 1. Consumption of video content is unaffected.	PASS
	3	CAFE interception when intra domain consumption between access network 222.222.222.0/24 CS and 202.202.202.0/24 CC	Packets intercepted by CAFEedge2 and forwarded to CAFEedge1 in domain 2. Consumption of video content is unaffected.	Packets intercepted by CAFEedge2 and forwarded to CAFEedge1 in domain 2. Consumption of video content is unaffected.	PASS
	4	CAFE interception when intra domain consumption between access network 202.202.202.0/24 CS and 222.222.222.0/24 CC	Packets intercepted by CAFEedge1 and forwarded to CAFEedge2 in domain 2. Consumption of video content is unaffected.	Packets intercepted by CAFEedge1 and forwarded to CAFEedge2 in domain 2. Consumption of video content is unaffected.	PASS
Inter Domain Content Forwarding	5	CAFE interception and forward when inter domain consumption between access network 101.101.101.0/24 CS and 202.202.202.0/24 CC	Packet flow: CAFEedge1 (domain1) -> CAFE forward (domain 1) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 2) -> CAFEedge1 (domain2). Consumption of video content is unaffected.	Packet flow: CAFEedge1 (domain1) -> CAFE forward (domain 1) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 2) -> CAFEedge1 (domain2). Consumption of video content is unaffected.	PASS

	6	CAFE interception and forward when inter domain consumption between access network 202.202.202.0/24 CS and 101.101.101.0/24 CC	Packet flow: CAFEedge1 (domain2) -> CAFE forward (domain 2) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 1) -> CAFEedge1 (domain1). Consumption of video content is unaffected.	Packet flow: CAFEedge1 (domain2) -> CAFE forward (domain 2) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 1) -> CAFEedge1 (domain1). Consumption of video content is unaffected.	**PASS**
	7	CAFE interception and forward when inter domain consumption between access network 111.111.111.0/24 CS and 222.222.222.0/24 CC	Packet flow: CAFEedge2 (domain1) -> CAFE forward (domain 1) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 2) -> CAFEedge1 (domain2). Consumption of video content is unaffected.	Packet flow: CAFEedge2 (domain1) -> CAFE forward (domain 1) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 2) -> CAFEedge1 (domain2). Consumption of video content is unaffected.	**PASS**
	8	CAFE interception and forward when inter domain consumption between access network 222.222.222.0/24 CS and 111.111.111.0/24 CC	Packet flow: CAFEedge2 (domain2) -> CAFE forward (domain 2) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 1) -> CAFEedge2 (domain1). Consumption of video content is unaffected.	Packet flow: CAFEedge2 (domain2) -> CAFE forward (domain 2) -> CAFE forward (domain 3-transit) -> CAFE forward (domain 1) -> CAFEedge2 (domain1). Consumption of video content is unaffected.	**PASS**

12.4 Test Results: Server awareness

Test Category	Test #	Test	Expected	Actual	Result
SNME	1	Consumption from same domain CC and CSes	CME terminal shows correct status for both CS and the correct CS selected.	CME terminal shows correct status for both CS and the correct CS selected.	PASS
	2	Consumption domain 1 (CC) and domain 2 (two CS, same content). Both CS status L	CME terminal shows status L for both CS (i.e. status 1) and CS selected random.	CME terminal shows status L for both CS (i.e. status L) and CS selected random.	PASS
	3	Consumption domain 1 (CC) and domain 2 (two CS, same content). 1 CS status L and other status M	CME terminal shows correct status for both CS (i.e. status 1 and 2) and CS at status L selected.	CME terminal shows correct status for both CS (i.e. status 1 and 2) and CS at status L selected.	PASS
	4	Consumption domain 1 (CC) and domain 2 (two CS, same content). 1 CS status L and other status H	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status L selected.	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status L selected.	PASS
	5	Consumption domain 1 (CC) and domain 2 (two CS, same content). 1 CS status M and other status H	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status M selected.	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status M selected.	PASS
	6	Consumption domain 1 (CC) and domain 2 (two CS, same content). 1 CS status M and other status D	CME terminal shows correct status for both CS (i.e. status 1 and 5) and CS at status M selected.	CME terminal shows correct status for both CS (i.e. status 1 and 5) and CS at status M selected.	PASS

	7	Consumption domain 1 (CC) and domain 3 (two CS, same content). Both CS status L	CME terminal shows status L for both CS (i.e. status 1) and CS selected random.	CME terminal shows status L for both CS (i.e. status L) and CS selected random.	**PASS**
	8	Consumption domain 1 (CC) and domain 3 (two CS, same content). 1 CS status L and other status M	CME terminal shows correct status for both CS (i.e. status 1 and 2) and CS at status L selected.	CME terminal shows correct status for both CS (i.e. status 1 and 2) and CS at status L selected.	**PASS**
	9	Consumption domain 1 (CC) and domain 3 (two CS, same content). 1 CS status L and other status H	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status L selected.	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status L selected.	**PASS**
	10	Consumption domain 1 (CC) and domain 3 (two CS, same content). 1 CS status M and other status H	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status M selected.	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status M selected.	**PASS**
	11	Consumption domain 1 (CC) and domain 3 (two CS, same content). 1 CS status M and other status D	CME terminal shows correct status for both CS (i.e. status 1 and 5) and CS at status M selected.	CME terminal shows correct status for both CS (i.e. status 1 and 5) and CS at status M selected.	**PASS**
	12	Consumption domain 2 (CC) and domain 3 (two CS, same content). Both CS status L	CME terminal shows status L for both CS (i.e. status 1) and CS selected random.	CME terminal shows status L for both CS (i.e. status L) and CS selected random.	**PASS**
	13	Consumption domain 2 (CC) and domain 3 (two CS, same content). 1 CS status L and other status M	CME terminal shows correct status for both CS (i.e. status 1 and 2) and CS at status L selected.	CME terminal shows correct status for both CS (i.e. status 1 and 2) and CS at status L selected.	**PASS**
	14	Consumption domain 2 (CC) and domain 3 (two CS, same content). 1 CS status L and other status H	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status L selected.	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status L selected.	**PASS**

	15	Consumption domain 2 (CC) and domain 3 (two CS, same content). 1 CS status M and other status H	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status M selected.	CME terminal shows correct status for both CS (i.e. status 1 and 3) and CS at status M selected.	PASS
	16	Consumption domain 2 (CC) and domain 3 (two CS, same content). 1 CS status M and other status D	CME terminal shows correct status for both CS (i.e. status 1 and 5) and CS at status M selected.	CME terminal shows correct status for both CS (i.e. status 1 and 5) and CS at status M selected.	PASS

12.5 Test Results: COMET system

Test Category	Test #	Test	Expected	Actual	Result
Intra-Domain	**1**	Publish static content in domain 1 and request from local CC	Content delivered to CC	Content delivered to CC	PASS
	2	Publish static content in domain 1 and request from local CC	Content delivered to CC	Content delivered to CC	PASS
Inter-Domain	**3**	Publish static content in domain 1 and request from CC domain 2	Content delivered to CC	Content delivered to CC	PASS
	4	Publish static content in domain 1 and request from CC domain 3	Content delivered to CC	Content delivered to CC	PASS

	5	Publish static content in domain 2 and request from CC domain 1	Content delivered to CC	Content delivered to CC	**PASS**
	6	Publish static content in domain 2 and request from CC domain 3	Content delivered to CC	Content delivered to CC	**PASS**
Non-static content	**7**	Publish non-static content (e.g. rtsp) in domain 1 and request from CC domain 2	Content delivered to CC	Content delivered to CC	**PASS**
	8	Publish non-static content in domain 1 and request from CC domain 3	Content delivered to CC	Content delivered to CC	**PASS**
	9	Publish non-static content in domain 2 and request from CC domain 1	Content delivered to CC	Content delivered to CC	**PASS**

13 Appendix D: User manuals

13.1 Streaming application

Since most of the configuration operations of the COMET entities (CRE/CME/SNME) are standard COMET operations, they are not described here (see D3.3 [4] for more information). The focus will be then placed on the operations to be performed in the elements outside direct COMET responsibility, name, CCs and CSs, highlighting the particularities of the specific pieces of Software used in the integration.

The needed pieces of SW are

- COMET CC SW on the CC (windows-based).
- VLC as streaming server on the CS (Linux-based/Ubuntu 10).
- SMA on the CS.

13.1.1 COMET Content Client SW on the CC

The steps for installing the Content Client in the user equipment are as follows:

1. Run the file 'COMET-CC.msi'. Agree license and click 'Next'.

Figure 20: COMET CC installation, License Screen

2. Enter CME host address and port. Click 'Next'.

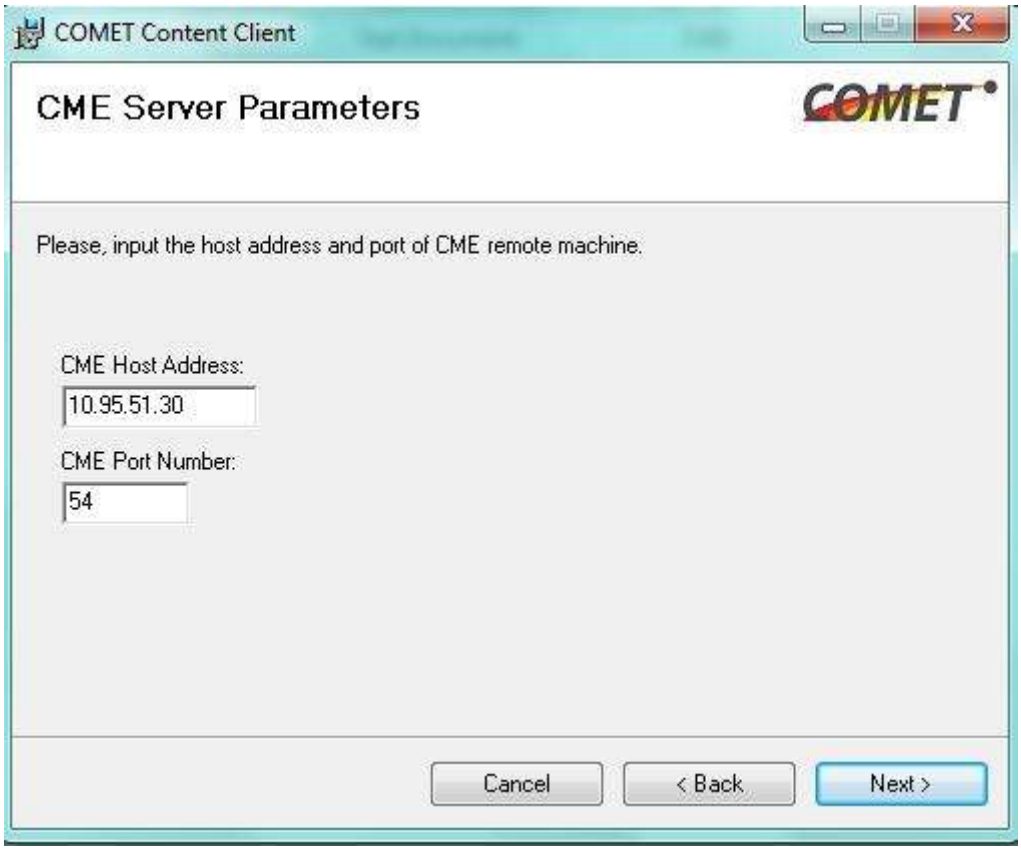

Figure 21: COMET CC installation, CME Configuration Screen

3. Enter the path where CC will be installed. Click Next.

Figure 22: COMET CC installation, Folder Selection Screen

4. Start installation. Click 'Next'.

Figure 23: COMET CC installation, Confirmation Screen

5. Installation process.

Figure 24: COMET CC installation, Process Screen

6. Installation finished. Click 'Close' to exit.

Figure 25: COMET CC installation, Final Screen

13.1.2 VLC as Streaming Server in the CS

VLC is available as a Linux Package. To install it in Ubuntu, just type the following two commands:

```
sudo apt-get update
sudo apt-get install vlc vlc-plugin-pulse mozilla-plugin-vlc
```

One VLC is installed, the http server can be started by typing the following command for IPv4

```
vlc my_content.mkv --sout
'#std{access=http,mux=ts,dst=server_address_ipv4:port/path/file.ts}'
```

or

```
vlc my_content.mkv --ipv6 --sout
'#std{access=http,mux=ts,dst=[server_address_ipv6]:port/path/file.ts}'
```

for IPv6, where `my_content.mkv` is the file to be streamed `server_address_ipv4` and `server_address_ipv6`, `port` is the port in the server where connections are accepted and `path/file.ts` is the path to the content.

In the case of RTSP streaming, and only for IPv4, the command would be

```
vlc my_content.mkv --sout '#rtp{dst=0.0.0.0,port=port,mux=ts,rtcp-
mux=1,sdp=rtsp://:port/content_path}'
```

where `my_content.mkv` is the file to be streamed, `port` is the both the port in the server where connections are accepted and the port in the client where the streaming is going to be received , while `content_path` is the path to the content.

13.1.3 SMA in CS

Before installing SMA in the CS, it is required to have first installed java and ant, which are available as Linux packages. To install them simply write the following two commands

```
sudo apt-get install ant
sudo apt-get install java-common
```

SMA is distributed in a compressed file format (.tar for linux, .zip for Windows). Copy the compress file to the server machine and follow the next steps:

- Uncompress the (.tar/.zip) file in a local directory.
- Open the ./src/ directory.
- Set up configuration parameters at file ./src/configuration.properties.
  - snmeHost: SNME address (IP address or host name).
  - snmePort: SNME port for incoming packets from SMA.
  - csHost: SMA address (IP address or host name).
  - csPort: SMA port for incoming packets from SNME.
  - cpuW1: Threshold range value for CPU (from 0 to cpuW1)
  - cpuW2: Threshold range value for CPU (from cpuW1 to cpuW2)
  - cpuW3: Threshold range value for CPU (from cpuW2 to cpuW3)
  - cpuMargin: Upper / Lower margin for CPU threshold ranges.
  - memW1: Threshold range value for MEM (from 0 to memW1)
  - memW2: Threshold range value for MEM (from memW1 to memW2)
  - memW3: Threshold range value for MEM (from memW2 to memW3)
  - memMargin: Upper / Lower margin for MEM threshold ranges.
  - bwW1: : Threshold range value for BW (from 0 to bwW1)
  - bwW2: : Threshold range value for BW (from bwW1 to bwW2)
  - bwW3: : Threshold range value for BW (from bwW2 to bwW3)
  - bwMargin: Upper / Lower margin for BW threshold ranges.
  - #Default Network Interface Speed (Mb/s)
  - defSpeed: Default Network Interface Speed (Mb/s). Mandatory required at Linux Machine.

After configuration, SMA can be launched by typing the command

ant run

13.2 VOD application

Since most of the configuration operations of the COMET entities (CRE/CME/SNME) are standard COMET operations, they are not described here (see D3.3 [4] for more information). The focus will be then placed on the operations to be performed in the elements outside direct COMET responsibility, name, CCs and CSs, highlighting the particularities of the specific pieces of software used in the integration.

The needed pieces of SW are

- COMET CC SW on the CC (windows-based)
- Apache Web server as VoD server on the CS (Linux-based/Ubuntu 10)
- SMA on the CS

13.2.1 COMET Content Client SW in the CC

Follow the same steps than in section 13.1.1.

13.2.2 Apache Web Server as VoD server in the CS

Apache Web Server version 2 is available as a Linux Package for installation in Ubuntu, in order to to that simply type

```
sudo apt-get update
sudo apt-get install apache2
```

Apache configuration files are typically installed at

```
/etc/apache2
```

in order to prepare the Apache Web Server for receiving connections both in IPv4 and IPv6, the following two lines has to be present in the file ports.conf

```
Listen 80
Listen [::]:80
```

Contents can be then placed at

```
 /var/www
```

where they can be served by the Apache Web Server. It is important to remark that video contents should be codified in MPEG Stream Format and identified by extensions .ts in order to enable proper treatment by the client VLC.

Now the video contents can be retrieved with the URL

http://server-IPv4/content.ts

for IPv4, or

http://[server-IPv6]/content.ts

for IPv6.

13.2.3 SMA in CS

Same than section 13.1.3

13.3 P2P application

Since most of the configuration operations of the COMET entities (CRE/CME/SNME/CAFE) are standard COMET operations, they are not described here (see D3.3 [4] for more information). The focus will be then placed on the operations to be performed in the elements outside direct COMET responsibility, name, CCs and CSs, highlighting the particularities of the specific pieces of software used in the integration.

The needed pieces of SW are

- COMET CC SW on the CC (windows-based).
- µTorrent as torrent Tracker in the CS (Windows-based).
- Apache Web Server as .torrent files repository in the CS.
- SMA on the CS.

13.3.1 COMET Content Client

Follow the same steps than in section 13.1.1.

13.3.2 Apache Web Server

In the P2P application, the Apache Web Server (version 2.2) is used as the repository for the .torrent containing the tracker information for the contents under distribution.

Apache2 is available as a windows installable package. Once installed, the configuration files are typically stored at

```
C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf
```

Or

```
C:\Program Files\Apache Software Foundation\Apache2.2\conf
```

Depending on the Windows version. There, inside the file httpd.conf, it has to be checked that it contains these two lines

```
Listen 80
Listen [::]:80
```

in order to enable IPv6 support.

Once Apache has been started .torrent files can be placed at

```
C:\Program Files (x86)\Apache Software Foundation\Apache2.2\htdocs
```

Or

```
C:\Program Files\Apache Software Foundation\Apache2.2\htdocs
```

Depending on the Windows version.

Now the .torrent file can be retrieved with the URL

http://server-IPv4/content.torrent

for IPv4, or

http://[server-IPv6]/content.torrent

for IPv6.

13.3.3 µTorrent as Torrent Tracker

µTorrent is available as a windows installer. Once µTorrent has been installed and launched the application has to be configured in order to act as a tracker.

For that go to Options->Preferences and in the Advanced section change the value of bt.enable_tracker to true

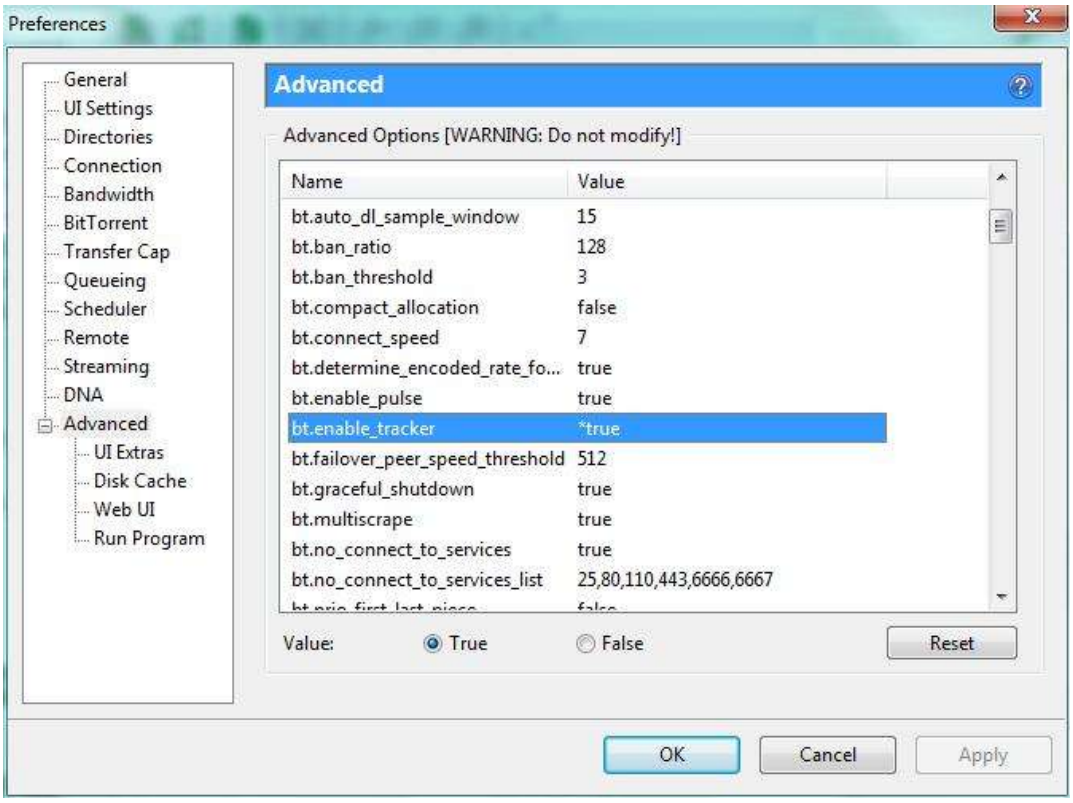

Figure 26: µTorrent, configuration as tracker

If IPv6 traffic is required in the Advanced the parameter net.disable_incoming_ipv6 has to be set to false.

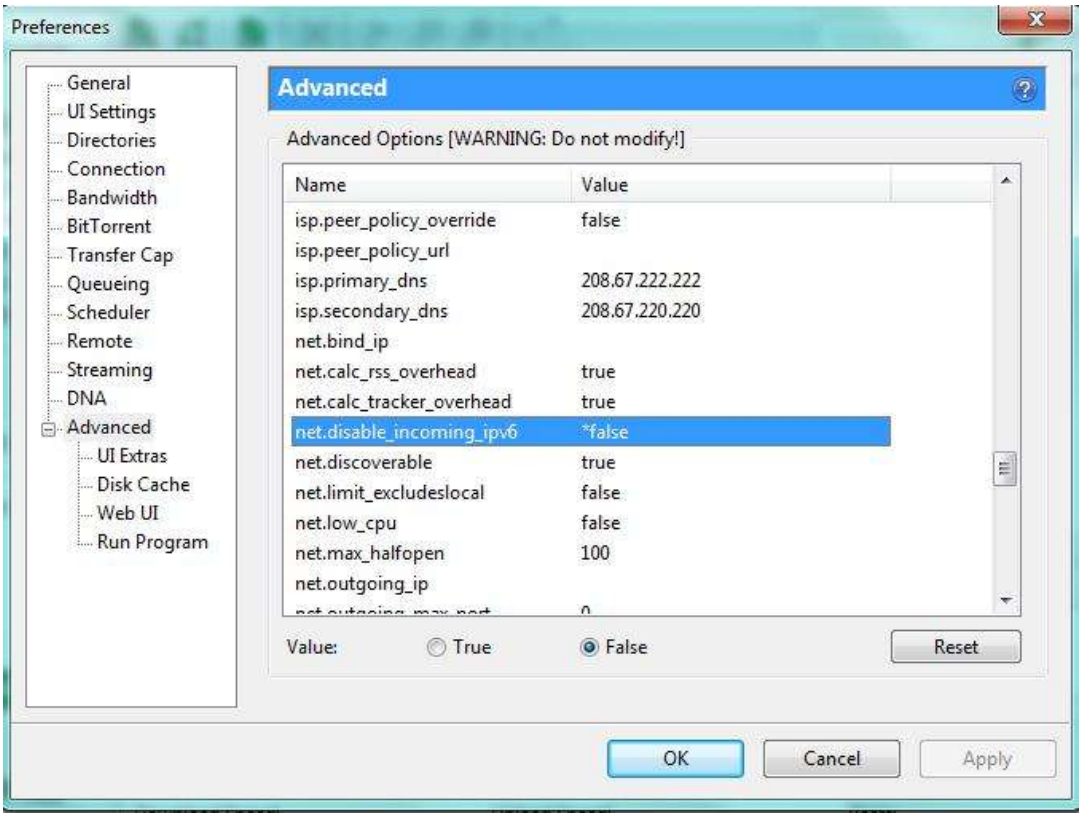

Figure 27: µTorrent, IPv6 activation

Last, µTorrent has to be configured to accept connections in a fixed port, the one that will be used for the tracker and will be specified in the .torrent file for the video content. Go to the 'Connection' section, choose a port in 'Port used for incoming connections' and remove check from option 'Randomize port each start'.

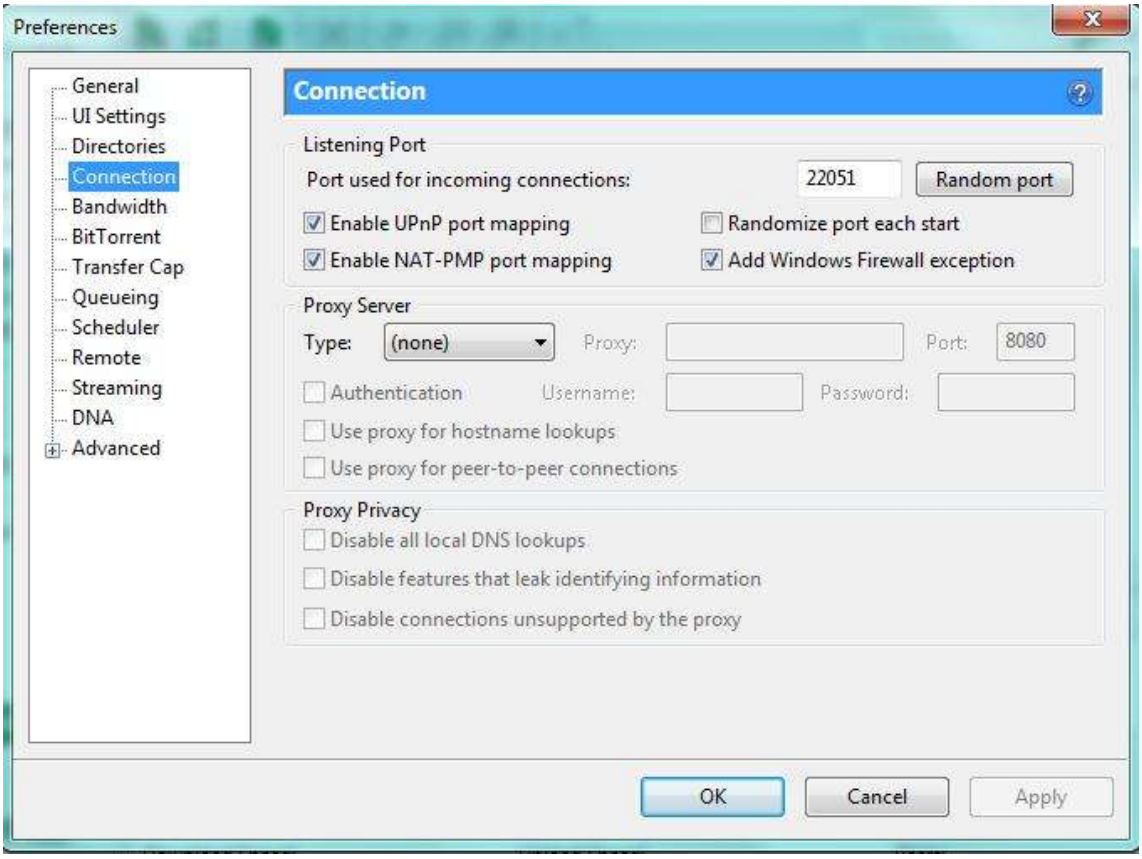

Figure 28: µTorrent, port configuration

Now the .torrent File for distributing a content via P2P can be created. In order to do that, select File→ Create New Torrent.

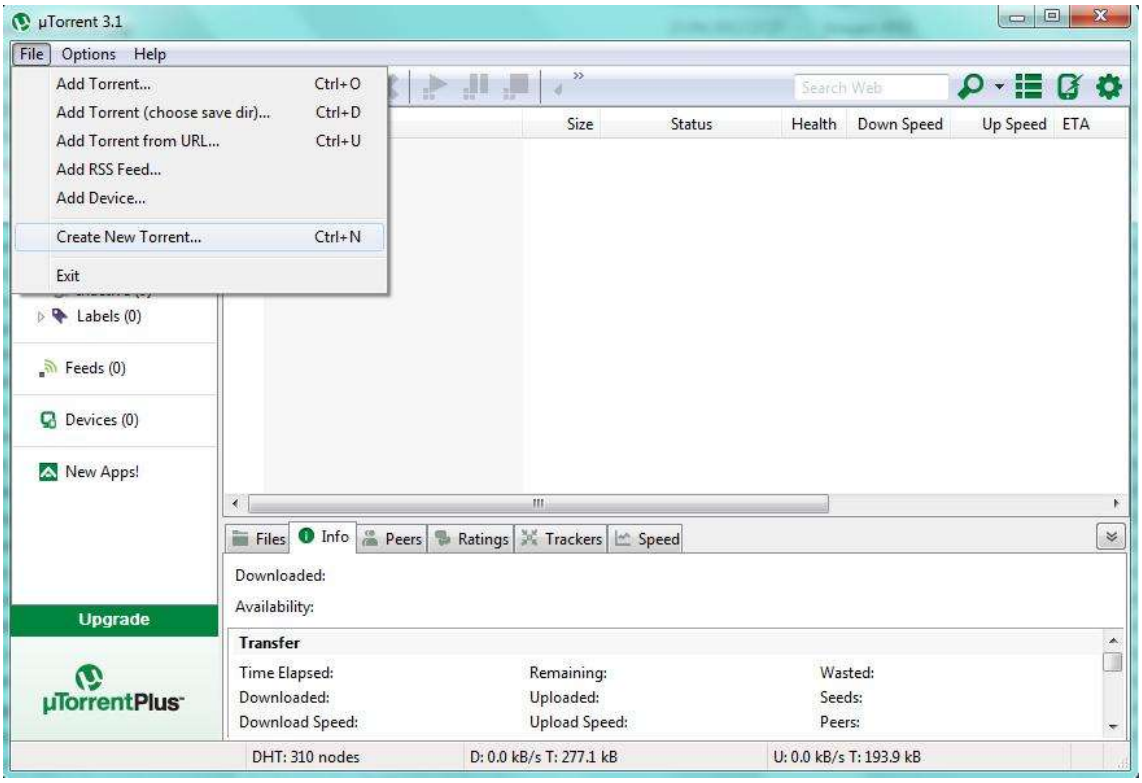

Figure 29: µTorrent, Torrent Creation I

At ‘Select Source’ add the content file. At section ‘Trackers’ enter the torrent tracker with this format: http://server_address:server_listening_port/announce where server_address is the IPv6/Ipv4 address of the server and server_listening_port is the port defined before for the tracker to listen.

Check the option ‘Start seeding’ at ‘Other’ section. Click ‘Create and save as….’. Save the torrent file.

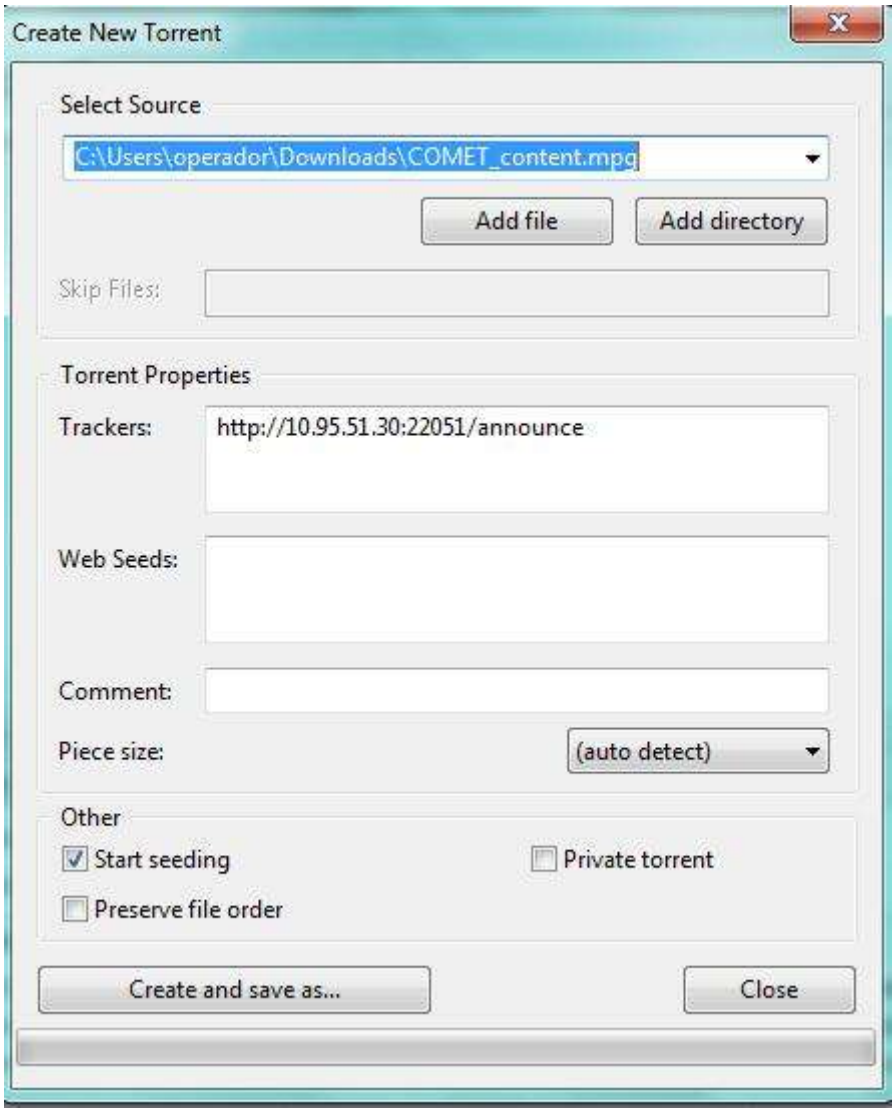

Figure 30: µTorrent, Torrent Creation 2

µTorrent will then start working both as tracker and initial seed for the selected content and any other µTorrent client can connect to the tracker and start content retrieval by using the .torrent file that has been generated.

13.3.4 SMA in CS

Same than section 13.1.3

13.4 Content streaming relay application

Since most of the configuration operations of the COMET entities (CRE/CME/SNME) are standard COMET operations, they are not described here (see D3.3 [4] for more information). The focus will be then placed on the operations to be performed in the elements outside direct COMET responsibility, name, CCs and CSs, highlighting the particularities of the specific pieces of software used in the integration.

The needed pieces of SW are

- COMET CC SW on the CC (windows-based).
- COMET CSS on CS with CSS function.
- COMET CSR on CS with CSR function.

- Python (tested with versions 2.6 and 2.7) on both CSs (CS with CSS function and CS with CSR function).
- SMA on both CSs (CS with CSS function and CS with CSR function).

13.4.1 COMET Content Client SW in the CC

Follow the same steps than in section 13.1.1.

13.4.2 COMET CSR on CS

In order to install CSR on CS just copy COMET CSR to CS's hard drive. Notice that CSR requires Python to be installed.

13.4.3 COMET CSR on CS

In order to install CSS on CS just copy COMET CSS to CS's hard drive. Notice that CSS requires Python to be installed.

13.4.4 Python on CS

Python is available as a Linux Package. To install it in Ubuntu, just type the following two commands:

```
sudo apt-get update
sudo apt-get install python
```

Notice that CSS and CSR have only been tested with python 2.6 and python 2.7. To see version of whcich version of Python is installed run following command:

```
python --version
```

13.4.5 SMA on CS

Same than section 13.1.3

14 Appendix E: Unit tests

14.1 CME

Self-contained tests for CME, were implemented using the JUnit framework and include tests for most CME packages, interfaces and functions. Figure 31 presents the results for JUnit tests for CME entity.

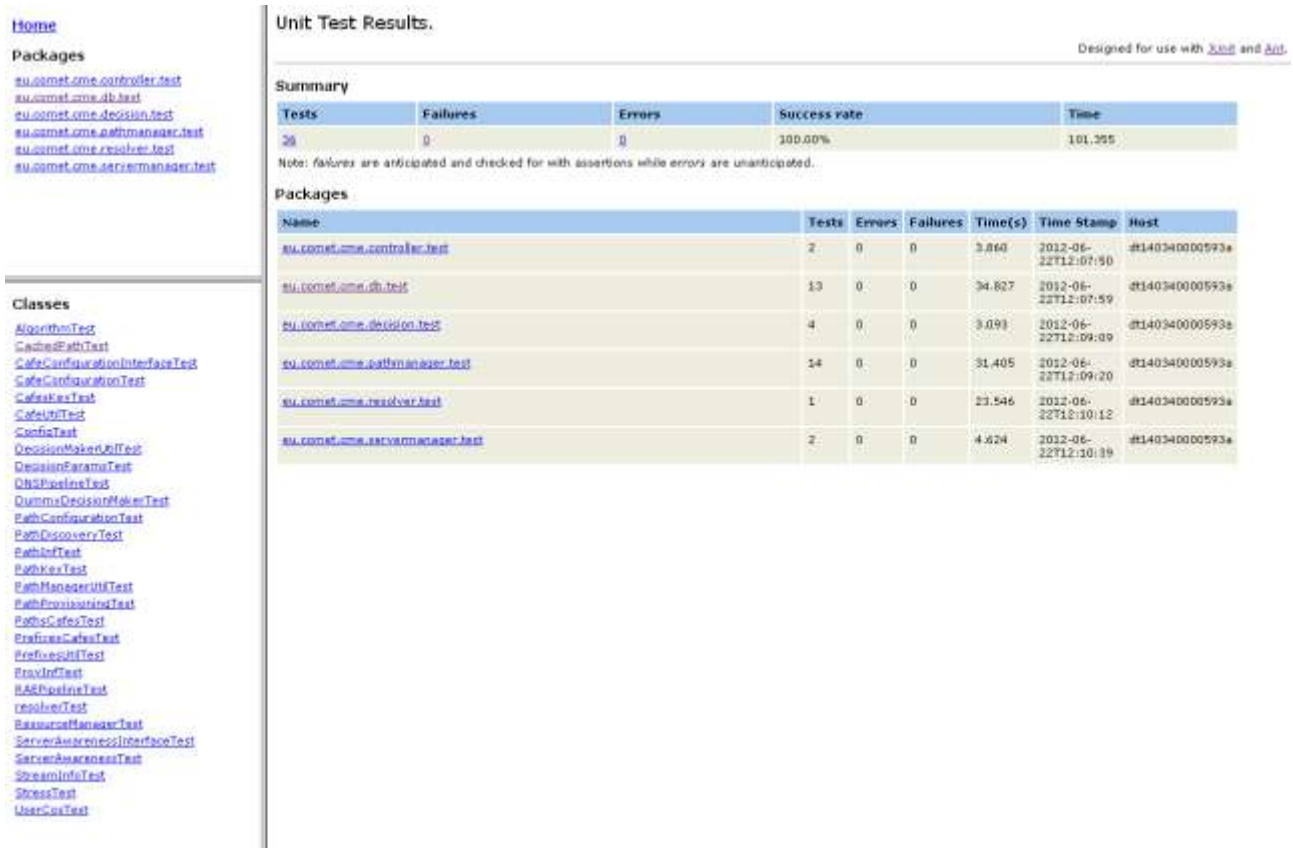

Figure 31: Junit test results for CME

The test classes are:

- `AlgorithmTest`: Tests for the decision algorithm
- `CachedPathTest`: Tests for all the queries of CachedPathDAO class
- `CafeConfigurationInterfaceTest`: Tests for the CME-CAFE interface
- `CafeConfigurationTest`: Tests for the CAFE configurator
- `CafesKeyTest`: Tests for all the queries of CafesKeyDAO class
- `CafeUtilTest`: Tests for all supporting functions used in CAFE configuration
- `ConfigTest`: Tests for all the queries of ConfigDAO class
- `DecisionMakerUtilTest`: Tests for all the supporting functions used in decision algorithm
- `DecisionParamsTest`: Tests for all the queries of DecisionParamsDAO class
- `DNSPipelineTest`: Tests for the CME-CC interface
- `DummyDecisionMakerTest`: Tests for the dummy decision algorithm
- `PathConfigurationTest`: Tests for the path configuration process
- `PathDiscoveryTest`: Tests for the path discovery process

- `PathInfTest`: Tests for all the queries of PathInfDAO class
- `PathKeyTest`: Tests for all the queries of PathKeyDAO class
- `PathManagerUtilTest`: Tests for the supporting functions used in path discovery, configuration and provisioning processes
- `PathProvisioningTest`: Tests for the path provisioning process
- `PathsCafesTest`: Tests for all the queries of PathsCafesDAO class
- `PrefixesCafesTest`: Tests for all the queries of PrefixesCafesDAO class
- `PrefixesUtilTest`: Tests for the longest prefix matching
- `ProvInfTest`: Tests for all the queries of ProvInfDAO class
- `RAEPipelineTest`: Tests for the CME-RAE interface
- `ResolverTest`: Tests for the CME-CRE interface
- `ResourceManagerTest`: Tests for the resource management
- `ServerAwarenessInterfaceTest`: Tests for the CME-SNME interface
- `ServerAwarenessTest`: Tests for the server awareness process
- `StreamInfoTest`: Tests for all the queries of StreamInfoDAO class
- `StressTest`: Stress testing of all processes
- `UserCosTest`: Tests for all the queries of UserCosDAO class