Project Number:	**FP7-257123**
Project Title:	**CONVERGENCE**
Deliverable Type:	**Report**
Dissemination Level	**Public**

Deliverable Number:	**D3.3**
Contractual Date of Delivery to the CEC:	**30.11.2011**
Actual Date of Delivery to the CEC:	**25.1.2012**
Title of Deliverable:	**Technologies and connectors infrastructure**
Workpackage contributing to the Deliverable:	**3**
Nature of the Deliverable:	**Report**
Editors:	**Mihai Tanase, Stefano Salsano, Richard Walker, Lucian Corlan**
Authors:	**Stefano Salsano, Andrea Detti, Giuseppe Tropea, Nicola Blefari Melazzi (CNIT), Leonardo Chiariglione (CEDEO), Helder Castro (INESC), Angelos – Christos Anadiotis, Aziz Mousas, Charalampos Patrikakis (ICCS), Thomas Huebner (Morpho), Mihai Tanase, Lucian Corlan (UTI), Panagiotis Gkonis (SIL), Jose Ribas, Daniel Sequeira (WIPRO)**
Keyword List:	**Architecture, protocols, components, services, applications, middleware, network, API, requirements, scalability, security, cross-concerns.**

Executive Summary

This document supersedes and replaces D3.2 (the previous deliverable from WP3), extending it in the following directions:

- Improved description of the distributed architecture of CONVERGENCE
- Improved description of the deployment approach (peers and nodes architecture)
- Improved description of the security mechanisms in CONVERGENCE
- Improved description of how scalability is incorporated in the architecture of the system
- Improved description of the VDI Naming Scheme

Unlike D3.2, this document no longer contains detailed descriptions of Technology Engines and their associated APIs. This information was included in D3.2 primarily to demonstrate technical progress by the implementation work packages that had no planned deliverables in year 1. Given that work on the detailed specification and implementation of some of the Technical Engines is still ongoing and that this is the last planned report from the System Architecture Work Package (WP3), the project has decided that this work will now be reported in future WP5 deliverables.

1 Introduction

The purpose of the CONVERGENCE system is to propose a novel content-centric framework that complements and enhances the current Internet architecture, building on and extending related MPEG standards. This report describes the technical concepts underlying the framework and its functional architecture, based on the use cases and functional requirements defined in Work Package 2 (WP2).

The report describes the following key features of the framework and their integration in the CONVERGENCE architecture.

- The 3-level architecture (Computing Platform, Middleware, Applications) of the system
- The Middleware (CoMid), (an extension of the standard MPEG middleware)
- The CONVERGENCE content-centric network (CoNet)
- CONVERGENCE security technology (CoSec)
- Extensions of the VDI format to support semantic linkage[1]
- The CONVERGENCE Publish/Subscribe mechanism
- The CONVERGENCE Digital Forgetting mechanism
- The ontology service.

To present these features, we provide different views of the architecture, at gradually increasing levels of technical detail.

- Overview of CONVERGENCE system (section 3)
    - High-level architecture diagrams
    - Main technical and architectural concepts
- Main technical and architectural concepts (section 4)
- The application level (section 5)[2]
- The middleware level (section 6)
- The computing platform level (section 7)

[1] The proposed format is an extension of the Digital Item (DI) format standardized by MPEG.

[2] A more detailed description of CONVERGENCE applications is provided in reports from WP7

2 Terms and Definitions

Term	Definition
Access Rights	Criteria defining who can access a VDI or its components under what conditions.
Advertise	Procedure used by a CoNet user to make a resource accessible to other CoNet users.
Application	Software, designed for a specific purpose that exploits the capabilities of the CONVERGENCE System.
Business Scenario	A scenario describing a way in which the CONVERGENCE System may be used by specific users in a specific context or, more narrowly, a scenario describing the products and services bought and sold, the actors concerned and, possibly, the associated flows of revenue in such a context.
CA	Central Authority
CCN	Content Centric Network
Cl_Auth_SC	Client Authentication with Smart Card (Challenge Response)
Cl_Auth_User_Pw	Client Authentication with Username and Password
Clean-slate architecture	The CONVERGENCE implementation of the Network Level, totally replacing existing IP functionality. See “Integration Architecture” and ”“Overlay Architecture” and “Parallel Architecture”.
CoApp	The CONVERGENCE Application Level.
CoApp Provider	A user providing Applications running on the CONVERGENCE Middleware Level (CoMid).
CoMid	The CONVERGENCE Middleware Level.
CoMid Provider	A user providing access to a single or an aggregation of CoMid services.

CoMid Resource	A virtual or physical object or service referenced by a VDI, e.g. media, Real World Objects, persons, internet services. It has the same meaning of "Resource" and it is used only to better specify the term "Resource" when there is a risk of a misunderstanding with the term "CoNet Resource".
Community Dictionary Service (CDS)	A CoMid Technology Engine that provides all the matching concepts in a user's subscription, search request and publication.
CoNet Provider	A user providing access to CoNet services, i.e. the equivalent of an Internet Service Provider.
CoNet Resource	A resource of the CoNet that can be identified by means of a name; resources may be either Named-data or a Named service access point.
Content-based resource discovery	A user request for resources, either through a subscription or a search request to the CONVERGENCE system (from literature). See "subscription" and "search".
Content-based Subscription	A subscription based on a specification of user's preferences or interests, (rather than a specific event or topic). The subscription is based on the actual content, which is not classified according to some predefined external criterion (e.g., topic name), but according to the properties of the content itself. See "Subscription" and "Publish-subscribe model".
Content-centric	A network paradigm in which the network directly provides users with content, and is aware of the content it transports, (unlike networks that limit themselves to providing communication channels between hosts).
CONVERGENCE Applications level (CoApp)	The level of the CONVERGENCE architecture that establishes the interaction with CONVERGENCE users. The Applications Level interacts with the other CONVERGENCE levels on behalf of the user.
CONVERGENCE Computing Platform level (CoComp)	The Computing Platform level provides content-centric networking (CoNet), secure handling (CoSec) of resources within CONVERGENCE and computing resources of peers and nodes.
CONVERGENCE Core	A semantic representation of the CoReST taxonomy.

Ontology (CCO)	See “CONVERGENCE Resource Semantic Type (CoReST)”
CONVERGENCE Device	A combination of hardware and software or a software instance that allows a user to access Convergence functionalities
CONVERGENCE Engine	A collection of technologies assembled to deliver specific functionality and made available to Applications and to other Engines via an API
CONVERGENCE Middleware level (CoMid)	The level of the CONVERGENCE architecture that provides the means to handle VDIs and their components.
CONVERGENCE Network (CoNet)	The Content Centric component of the CONVERGENCE Computing Platform level. The CoNet provides access to named-resources on a public or private network infrastructure.
CONVERGENCE node	A CONVERGENCE device that implements CoNet functionality and/or CoSec functionality.
CONVERGENCE peer	A CONVERGENCE device that implements CoApp, CoMid, and CoComp (CoNet and CoSec) functionality.
CONVERGENCE Resource Semantic Type (CoReST)	A list of concepts or terms that makes it possible to categorize a resource, establishing a connection with the resource’s semantic metadata.
CONVERGENCE Security element (CoSec)	A component of the CONVERGENCE Computing Platform level implementing basic security functionality such as storage of private keys, basic cryptography, etc.
CONVERGENCE System	A system consisting of a set of interconnected devices - peers and nodes - connected to each other built by using the technologies specified or adopted by the CONVERGENCE specification. See “Node” and “Peer”.
Dec_Key_Unwrap	Key Unwrapping and Content Decryption
DIDL	Digital Item Description Language
Digital forgetting	A CONVERGENCE system functionality ensuring that VDIs do not remain accessible for indefinite periods of time, when this is not the intention of the user.

Digital Item (DI)	A structured digital object with a standard representation, identification and metadata. A DI consists of resource, resource and context related metadata, and structure. The structure is given by a Digital Item Declaration (DID) that links resource and metadata.
Domain ontology	An ontology, dedicated to a specific domain of knowledge or application, e.g. the W3C Time Ontology and the GeoNames ontology.
Elementary Service (ES)	The most basic service functionality offered by the CoMid.
Enc_Key_Wrap	Encryption and Key Wrapping
Entity	An object, e.g. VDIs, resources, devices, events, group, licenses/contracts, services and users, that an Elementary Service can act upon or with which it can interact.
Expiry date	The last date on which a VDI is accessible by a user of the CONVERGENCE System.
Fractal	A semantically defined virtual cluster of CONVERGENCE peers.
Group_Sig	Group Signature
ICN	Information Centric Network
Identifier	A unique signifier assigned to a VDI or components of a VDI.
Integration Architecture	An implementation of CoNet designed to integrate CoNet functionality in the IP protocol by means of a novel IPv4 option or by means of an IPv6 extension header, making IP content-aware. See “Clean-state Architecture”, “Overlay Architecture”, “Parallel Architecture”
IP	Identity Provider
License	A machine-readable expression of Operations that may be executed by a Principal.
Local named resource	A named-resource made available to CONVERGENCE users through a local device, permanently connected to the network. Users have two options to make named-resources available to other users: 1) store the resource in a device, with a permanent connection to the

	network; 2) use a hosting service. In the event she chooses the former option, the resource is referred to as a local named-resource.
Metadata	Data describing a resource, including but not limited to provenance, classification, expiry date etc.
MPEG eXtensible Middleware (MXM)	A standard Middleware specifying a set of Application Programming Interfaces (APIs) so that MXM Applications executing on an MXM Device can access the standard multimedia technologies contained in the Middleware as MXM Engines.
MPEG-M	An emerging ISO/IEC standard that includes the previous MXM standard.
Multi-homing	In the context of IP networks, the configuration of multiple network interfaces or IP addresses on a single computer.
Named resource	A CoNet resource that can be identified by means of a name. Named-resources may be either data (in the following referred to as “named-data”) or service-access-points (“named-service-access-points”).
Named service access point	A kind of named-resource, consisting of a service access point identified by a name. A named-service-access-point is a network endpoint identified by its name rather than by the Internet port numbering mechanism.
Named-data	A named-resource consisting of data.
Network Identifier (NID)	An identifier identifying a named resource in the CONVERGENCE Network. If the named resource is a VDI or an indentified VDI component, its NID may be derived from the Identifier (see “Identifier”).
Overlay architecture	An implementation of CoNet as an overlay over IP. See “Clean-state Architecture” and “Integration Architecture” and “Parallel Architecture”
Parallel architecture	An implementation of CoNet as a new networking layer that can be used in parallel to IP. See “Clean-state Architecture” and “Integration Architecture” and ””Overlay Architecture”
PKI	Public Key Infrastructure

Policy routing	In the context of IP networks, a collection of tools for forwarding and routing data packets based on policies defined by network administrators.
Principal (CoNet)	The user who is granted the right to use a *CoNet Principal Identifier* for naming its named resources. For example, the principal could be the provider of a service, the publisher or the author of a book, the controller of a traffic lights infrastructure, or, in general, the publisher of a VDI. A Principal may have several Principal Identifiers in the CoNet.
Principal (Rights Expression Language)	The User to whom Permissions are Granted in a License.
Principal Identifier (CoNet)	The Principal identifier is a string that is used in the Network Identifiers (NID) of a CoNet resource, when the NID has the form: NID = <namespace ID, hash (Principal Identifier), hash (Label)> In this approach, hash (Principal Identifier) must be unique in the namespace ID, and Label is a string chosen by the principal in such a way that hash(Label) is unique for in the context of the Principal Identifier.
Publish	The act of informing an identified subset of users of the CONVERGENCE System that a VDI is available.
Publisher	A user of CONVERGENCE who performs the act of publishing.
Publish-subscribe model	CONVERGENCE uses a content-based approach for the publish-subscribe model, in which notifications about VDIs are delivered to a subscriber only if the metadata / content of those VDIs match constraints defined by the subscriber in his Subscription VDI.
Real World Object	A physical object that may be referenced by a VDI.
REL	Rights Expression Language
Resource	A virtual or physical object or service referenced by a VDI, e.g. media, Real World Objects, persons, internet services.
Scope (in the context of routing)	In the context of advertising and routing, the geographical or administrative domain on which a network function operates (e.g. a well defined section of

	the network - a campus, a shopping mall, an airport -, or to a subset of nodes that receives advertisements from a service provider).
Search	The act through which a user requests a list of VDIs meeting a set of search criteria (e.g. specific key value pairs in the metadata, key words, free text etc.).
Serv_Auth	Server Authentication without Smart Card
Service Level Agreement (SLA)	An agreement between a service provider and another user or another service provider of CONVERGENCE to provide the latter with a service whose quality matches parameters defined in the agreement.
Sig	Signature
Smart_Card Role_Auth_SC	Role Authentication towards Smart Card
SP	Service Provider
Subscribe	The act whereby a user requests notification every time another user publishes or updates a VDI that satisfies the subscription criteria defined by the former user (key value pairs in the metadata, free text, key words etc.).
Subscriber	A user of CONVERGENCE who performs the act of subscribing.
Timestamp	A machine-readable representation of a date and time.
Tool	Software providing a specific functionality that can be re-used in several applications.
Trials	Organized tests of the CONVERGENCE System in specific business scenarios.
Un-named-data	A data resource with no NID.
Us_Reg_IP	User Registration to Identity Provider
Us_Reg_SP	User Registration to Service Provider
User	Any person or legal entity in a Value-Chain connecting (and including) Creator and End-User possibly via other Users.
User (in OSI sense)	In a layered architecture, the term is used to identify an entity exploiting the

	service provided by a layer (e.g. CoNet user).
User ontology	An ontology created by CONVERGENCE users when publishing or subscribing to a VDI.
User Profile	A description of the attributes and credentials of a user of the CONVERGENCE System.
Versatile Digital Item (VDI)	A structured, hierarchically organized, digital object containing one or more resources and metadata, including a declaration of the parts that make up the VDI and the links between them.

3 Overview of CONVERGENCE system

The goal of the CONVERGENCE project is to enhance the Internet with a content-centric [3], publish-subscribe service model [19], based on a common container for any kind of digital data, including representations of people and Real World Objects. We call this container the Versatile Digital Item (VDI). VDIs are the basic unit of distribution and transaction in the CONVERGENCE network. The definition of VDIs is derived from the MPEG-21 Digital Item Declaration Standard [22].

VDIs can incorporate any kind of information, including information required for signalling and control, and therefore minimize the need to store external information and states outside the data unit (though this is still allowed). The introduction of VDIs marks a shift from "host-centric" to "content-centric" networking, that is to a form of networking in which the network layer provides users with content, instead of providing communication channels between hosts, and is aware of this content, at least in the sense it knows its name. This shift is analogous to the switch from circuit to packet switching: in circuit switching a PCM slot contains only user data; in packet switching an IP datagram also contains (among other things) destination addresses. Similarly, in "content switching", the VDI contains a complete package of user data and meta-data describing content and how to handle it.

The VDI container can be used to encapsulate any kind of digital information: not only classical media files, but also data about services, people and Real World Objects (RWOs) (e.g. items of merchandise identified with an RFID). VDIs bind meta-information (describing the content and structure of the item) and resources (other VDIs, audio, images, video, text, descriptors of RWOs, descriptors of people etc.). The meta-data describing the VDI includes structural information, describing the content of the VDI; cryptographic keys allowing robust authentication and protection of information included in the VDI; rights information defining rights to use the item; and an expiry date, supporting "digital forgetting". VDIs are identified by a unique identifier, which is translated to a network-level name used to route the VDI. It can also be the same as the network-level name.

The second key feature of CONVERGENCE is support for a publish/subscribe service model: subscribers register their interest in an event, or a pattern of events, and are asynchronously notified of events generated by publishers. Publish/subscribe effectively decouples the application end-points in space, time and synchronization. This allows for greater scalability, a more dynamic network topology and a much enlarged and flexible typology of services.

Figure 1 depicts the main players in the CONVERGENCE framework: publishers advertise *resources* (data and service-access-points) on the CONVERGENCE system; *subscribers* express their interest in specific resources. The system notifies subscribers when the resources become available. Users can also *search* for resources and obtain an immediate response. In this respect *search* can be seen as a sub case of subscribe.

VDIs are used both to publish and to subscribe to content. Every resource that is stored or published in the CONVERGENCE system is associated with a VDI. Subscriptions express criteria that can be verified by inspecting VDI information. Therefore, CONVERGENCE system supports *content-based* subscriptions as defined in [19].

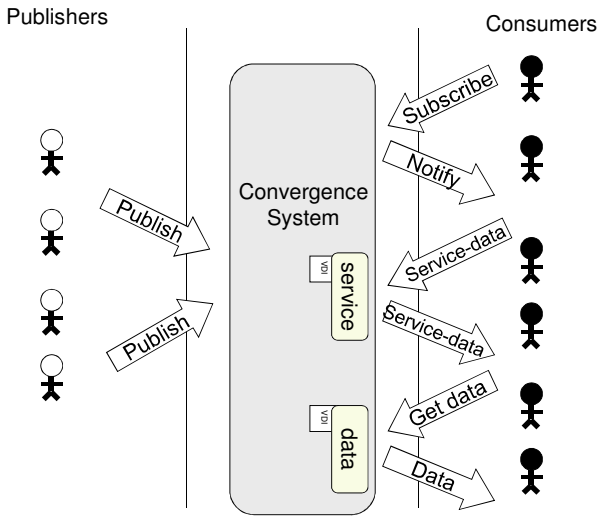


Figure 1: Service model

3.1 The CONVERGENCE Architecture

CONVERGENCE specifies an Information and Communication Technology (ICT) environment with the features described in this report. A CONVERGENCE system is an implementation of the specification consisting of a set of interconnected **peers** and **nodes** collectively called "CONVERGENCE devices".

Peers are based on a 3-level architecture. From the top down:

1. Application level (CoApp) –optional component
2. Middleware level (CoMid) –mandatory component
3. Computing Platform level (CoComp) –mandatory component

The Computing Platform level comprises key functional blocks providing novel content-centric networking (CoNet) and secure handling (CoSec) of resources within CONVERGENCE. The Computing Platform level also provides interfaces to access the local resources of the CONVERGENCE peers.

Nodes are devices that only include a CoNet networking component and/or a CoSec component, belonging as in peers, to the Computing Platform level. Therefore we can have CoNet nodes and CoSec nodes (see Figure 2).

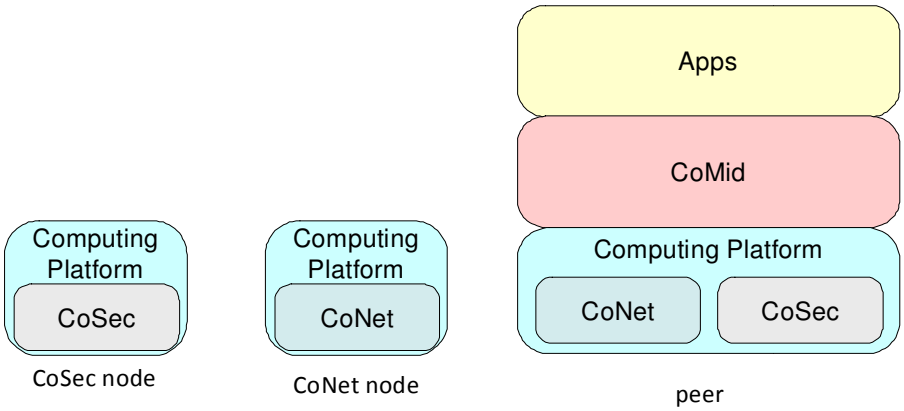


Figure 2: CONVERGENCE nodes and peers

Figure 3 depicts the three conceptual levels of the CONVERGENCE system architecture, summarizing the scope of each level and the kinds of information exchanged at the interfaces between levels.

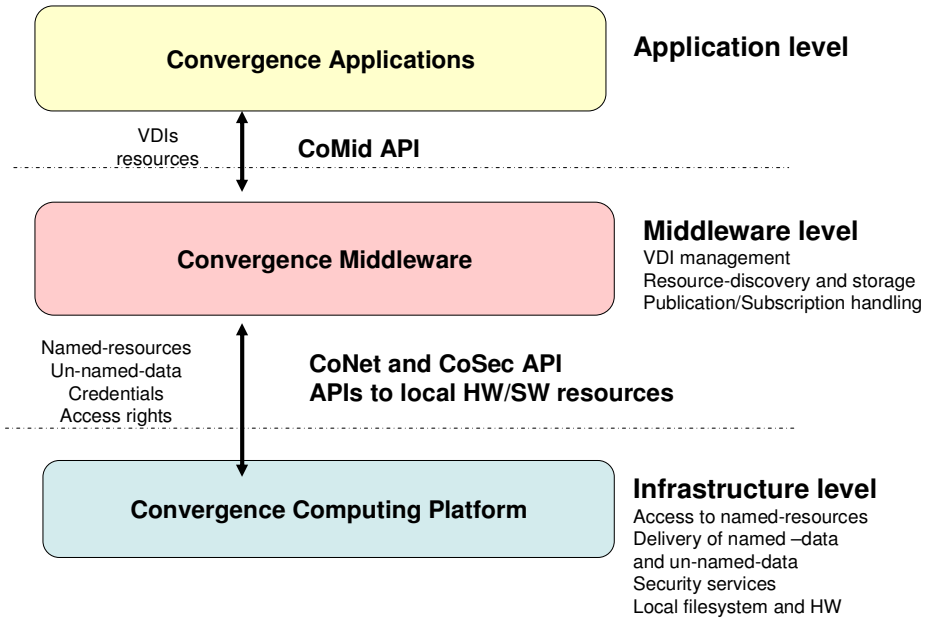


Figure 3: Overview of CONVERGENCE architectural levels

Figure 4 depicts the complete architecture of a peer. Each of the 3 levels has its own structure and communicates with other levels via standard APIs.

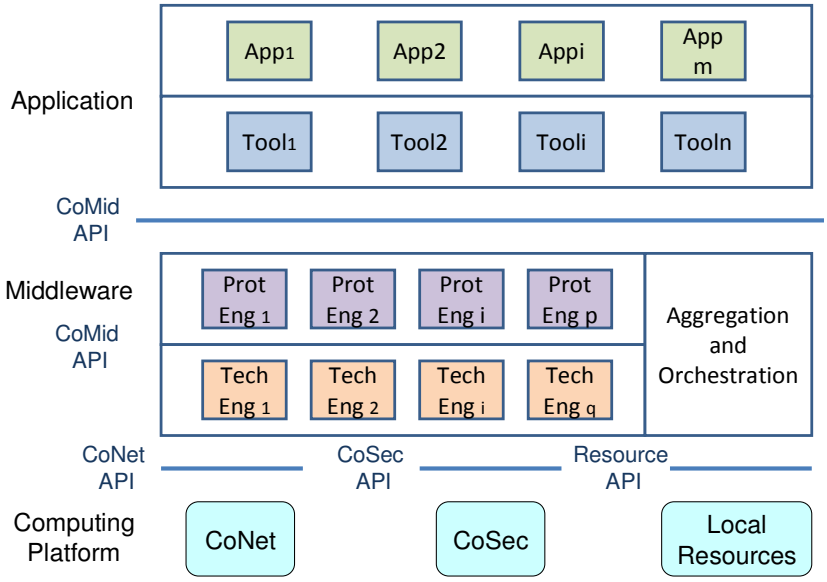


Figure 4: Architecture of a CONVERGENCE peer

3.1.1 Applications

Applications provide users of CONVERGENCE peers with the means to create, process and consume VDIs and their components, and thus to flexibly manipulate digital resources and deal with physical objects and their digital counterparts, as well as with people and their digital identities. These resources are produced and managed at the Application level. CONVERGENCE-compliant Applications use services provided by the CONVERGENCE middleware to create descriptors of resources in the form of VDIs. Hence resources and VDIs are exchanged at the interface between Applications and Middleware.

Users can download/install Applications to a peer. Applications can activate local and remote middleware functionalities, via the standard CoMid API. To expedite the development of Applications, CONVERGENCE defines special re-usable Application elements called Tools. Tools facilitate re-use of code in Applications; an Application can make use of several tools. The CONVERGENCE applications level can be seen as split into two sub-levels: a User Applications sub-level and a Tools sub-level. Therefore the term Application can be used in a wider sense to refer to both User Applications and Tools.

3.1.2 Middleware

The CONVERGENCE Middleware (CoMid) is the level responsible for creating, retrieving, manipulating and consuming VDIs. VDIs are published in the CoMid. The CoMid allows users to search for them using semantic search operations and delivers the results.

The CONVERGENCE Middleware level is based on the MPEG-M standard [21], which provides a distributed eco-system of Engines, i.e. components that can be activated by Applications running on MPEG-M compliant devices. CONVERGENCE builds on and expands the MPEG-M design. Specifically, CONVERGENCE introduces new Engines to:

- Implement the publish/subscribe paradigm on top of MPEG-M Event Notification services.
- Allow semantic searching of and subscriptions to published resources.

The CoMid is thus composed of Engines. Many are "native" MPEG-M engines, some are extended MPEG-M engines and a few are new, CONVERGENCE-specific, Engines.

There are two types of Engines:

1. Protocol Engines (PE) that activate functionalities in remote or local peers
2. Technology Engines (TE) that are typically called by PEs to execute specific functionalities.

Annex 1 provides a list of PEs and TEs included in CONVERGENCE along with the respective definitions.

An Application call may involve more than one PE/TE. Therefore CONVERGENCE provides a standard mechanism for "aggregating" PEs and "orchestrating" TEs. Work is under way to develop a single aggregation/orchestration technology applicable to PEs, TEs and, potentially, to Applications as well. Engines communicate via the CoMid API.

Figure 5 depicts how an Application call may involve a chain of PEs/TEs. However, chains need not be linear: where useful, Applications can directly call TEs without intermediation by a PE.

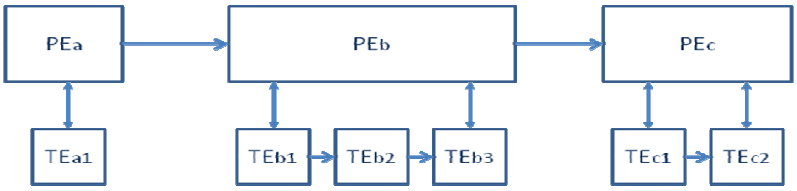


Figure 5: A chain of Protocol and Technology Engines

Figure 6 depicts how Peer 1 talks to Peer 2 via an internal PE, triggering the establishment of a chain of PEs/TEs.

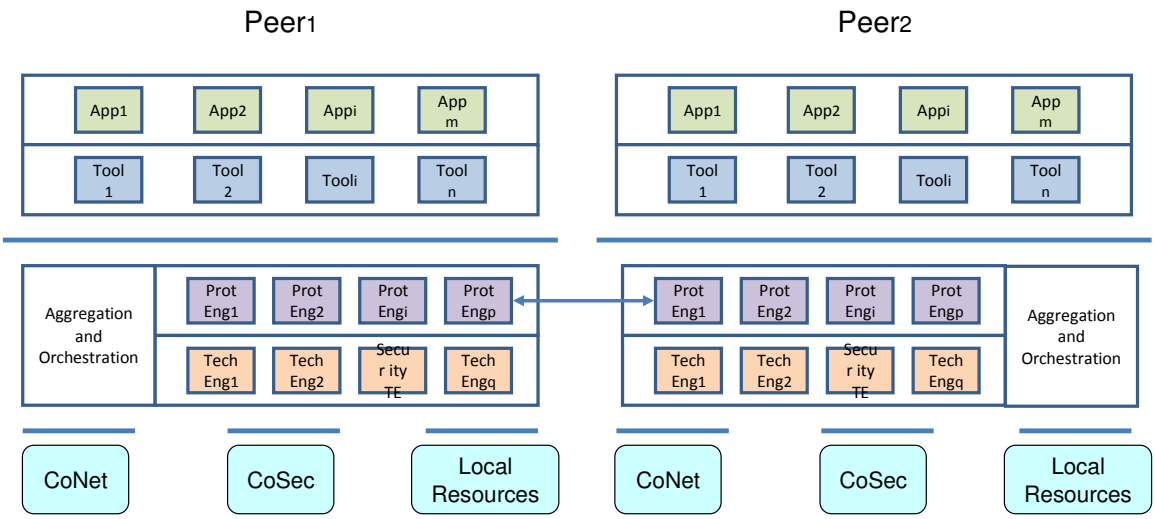


Figure 6: Communication between peers via PEs

CoMid provides the interface between Applications and the two key Computing Platform components: CoNet and CoSec, via their counterparts at the middleware level, the CoNet TE and the Security TE.

3.1.3 Computing Platform

The CONVERGENCE Computing Platform hosts specialized network and security modules, as well as interfaces to local resources, such as file-systems and processing power.

Engines access the local resources of the computing platform via a Computing Platform-dependent API.

When resources are stored inside the network they became network named-resources (or interactive data sessions). The CoNet communication functionalities are accessed, at a low-level, via the CoNet API. Applications access these functionalities via the CoNet middleware engine.

Credentials and access-rights, extracted from Licenses embedded in VDIs, are exchanged with the CoSec components of the CONVERGENCE-enabled Computing Platform. Low-level security functionalities in the CoSec block are accessed via the CoSec API. Applications access these functionalities via the Security middleware engine.

3.1.4 A distributed view of the CONVERGENCE system

Figure 7 provides a distributed view of the CONVERGENCE system, seen as a set of interconnected peer and CoNet/CoSec nodes. Application functionalities (and related security) are provided in a distributed way by CONVERGENCE peers. Network level functionality and security functionality are realized in a distributed way, exploiting CoNet/CoSec nodes.

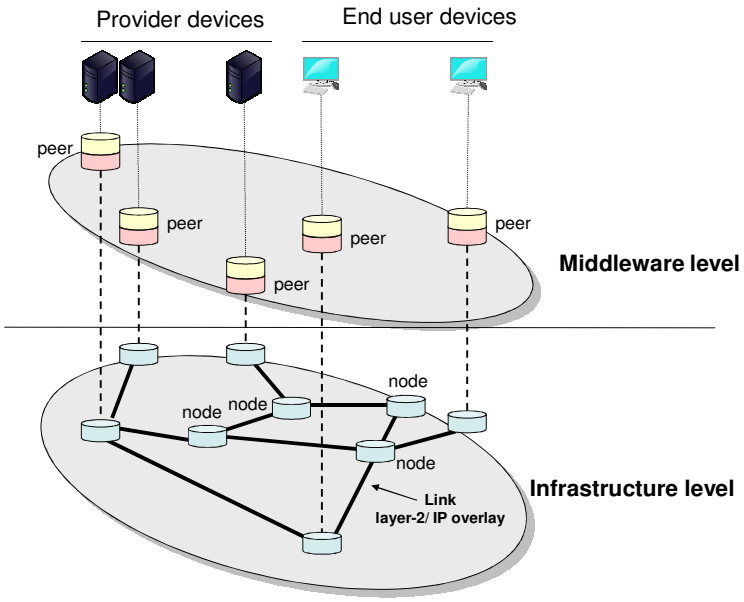


Figure 7: Distributed view of CONVERGENCE system

3.1.5 Deployment architecture

3.1.5.1 Peer Architecture

As already indicated in Figure 4, a CONVERGENCE peer consists of the middleware and the network layers, and, in some cases, an application layer. This means it has to include the software and hardware components required to support the middleware and network protocol and technologies.

As described in deliverable D6.1, the CONVERGENCE middleware is developed in the Java programming language and, as such, requires a Java Virtual Machine (JVM). Our implementation is based on JVM version 5 and has also been tested with version 6. This means that all peers require a JVM. Depending on the engines they host they may also require the BouncyCastle library for Java security extensions, and the Enterprise Java Beans (EJB) framework. This supports the Tomcat or JBoss application servers, which host elementary services and applications, as well as smart card hardware and software components in the security module.

Figure 8 depicts the deployment architecture for a peer containing:

- Protocol and/or Technology Engines
- Elementary Services

The diagram shows a gap between the application layer and the other layers. This is because some peers do not have an application layer: they are just elementary or aggregated Service Providers. In other words, the core of the peer is the middleware and the computing platform.

Peers can run on Windows (the current implementation has been tested on Windows Vista and Windows 7), Linux (tested with kernel versions 2.6.35.X) and Mac OS (tested with version OSX 10.6). In the next stages of development, we will also test our implementation on mobile platforms (e.g. Android).

CONVERGENCE supports COSEC (and smart cards) and CONET at the computing platform level. However, the computing platform can operate without these modules. At the time of writing, the full implementation of CONET is available only for Linux. However, COMID provides a CONET Technology Engine which provides an abstraction layer for the CONET on different computing platforms and can also work without an implementation of the CONET, simulating its functionality at the middleware level, without changing any of its features.

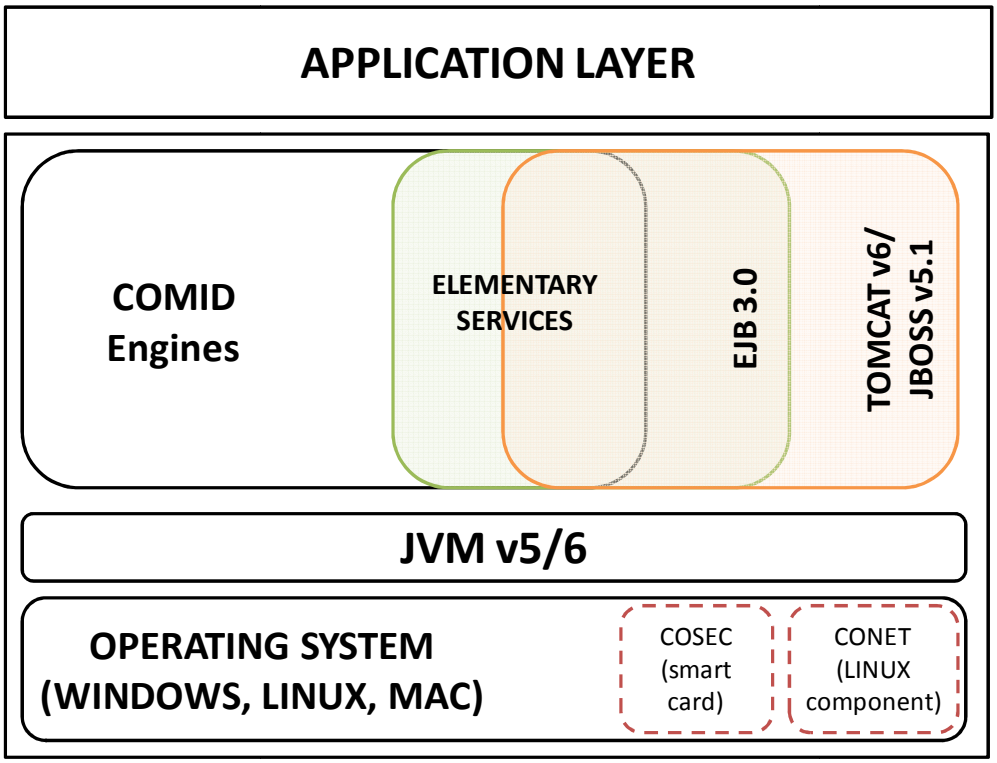


Figure 8 - Peer Deployment Architecture

Apart from EJB3 (Enterprise JavaBean3), Figure 8 does not depict the additional libraries/frameworks required by specific engines. These are specified in the engines' own dependencies. By contrast, the EJB3 framework is a dedicated component in the peer deployment architecture, used not only by

elementary services (EJB3 API), but also by the elementary/aggregated services provider (EJB3 runtime libraries). EJB3 and the application servers (Tomcat v6/JBoss v5.1) are required whenever the peer is a Service Provider. In some configurations, they may also be exploited by applications, which can use them to provide functionality for web applications.

The hardware and connectivity requirements of a full CONVERGENCE peer depend on the number of services it has to provide and the frequency of the incoming requests and cannot be captured deterministically. Based on the requirements of the VM, and the application servers, we estimate that the minimum requirements for a normal configuration are:

- CPU: >=2 cores, >=2.1 GHz/core
- RAM: >= 4GB
- Storage: >= 10GB (minimum OS)

However, configurations that only support a small number of Technology Engines may have lower requirements. If a peer contains only the security engine and use elementary services to perform other operations, it could as well be a low cost laptop or a mobile device. All that is required is that it should contain the software required to run the local engines (JVM, libraries, etc).

3.1.5.2 Node Architecture

In this section, we describe the functional architecture of a CONET node. For descriptions of the implementation of individual software modules, readers are referred to deliverable D6.1.

As previously presented (D5.1, R1.2), CONET has five types of nodes: border-nodes, internal-nodes, end-nodes, serving-nodes and Name Routing System Nodes. In what follows we describe their functional architectures one by one.

Border-Node

Figure 9 depicts the functional architecture of a border node. A border node performs three high level functions: forwarding carrier-packets across the CONET, caching named-data CIUs, verifying cached named-data CIUs.

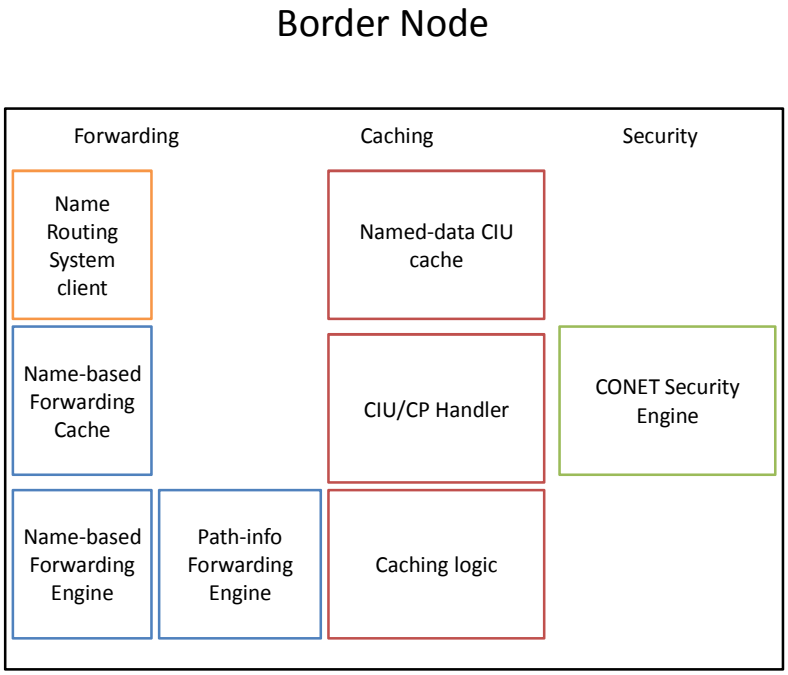


Figure 9 – Functional architecture of a border node

A carrier-packet may contain an Interest CIU, a segment of a named-data CIU, or an unnamed-data item.

The Name-Based Forwarding Engine handles forwarding of carrier-packets that contain an Interest CIU. The engine checks the availability of the requested data in the local cache. If the request content is not in the cache, the engine selects the next CONET hop, looking it up in the Name-Based Forwarding Cache. A forwarding entry in the Name-Based Forwarding Cache contains the mapping between a NID and the CSS address (e.g. IP address) of the next border-node toward the serving-node. If the required forwarding entry is not present in the cache, the Name Routing System Client requests the entry from a Name Routing System Node. The entry is then inserted in the Name-Based Forwarding Cache and the least recently used entry is removed (assuming to use the so-called lookup-and-cache approach; other route cache policies are possible and under investigation).

The Path-Info Forwarding Engine handles forwarding of carrier-packets that contain segments of named-data CIUs. Forwarding is based on the path-info field contained in the carrier-packet.

Forwarding of carrier-packets that contain unnamed-data is carried out by the Name-based Forwarding Engine when the packets contain the NID, or by the Path-info Forwarding Engine when the packets only specify the path-info field.

The caching processes store named-data CIUs and deliver segments of cached named-data CIUs. All caching processes are controlled by the Caching Logic.

To store a named-data CIU, a copy of carrier-packets containing named-data CIU segments is transferred by the Path-info Forwarding Engine to the Caching Logic, which may choose to cache the related CIU according to a specific policy. If the caching logic chooses to cache a traversing named-data CIU, the CIU/CP (Carrier Packet) function handles the reassembly process that reconstructs a CIU from the segments contained in carrier-packets traversing the node. When the CIU is reassembled, it is stored in the cache. From this time onwards, the caching logic takes care of periodically monitoring the expiration time of the cached named-data CIU and the match to the version in the origin serving-node.

To deliver a segment of a cached named-data CIU, the caching logic instructs the CIU/CP function to fetch the segment from the cache and to insert it in an outgoing carrier-packet.

Before caching a named-data CIU, the caching logic uses the Security Engine to check its validity. Security checks are based on the security fields contained in the CIU.

Internal-Node

Figure 10 depicts the functional architecture of an internal-node. An internal-node performs the same functions as a border node except forwarding of carrier-packets. This function is performed by lower level technologies: e.g. an IP modules that can handle CONET option in the IP (see [46]).

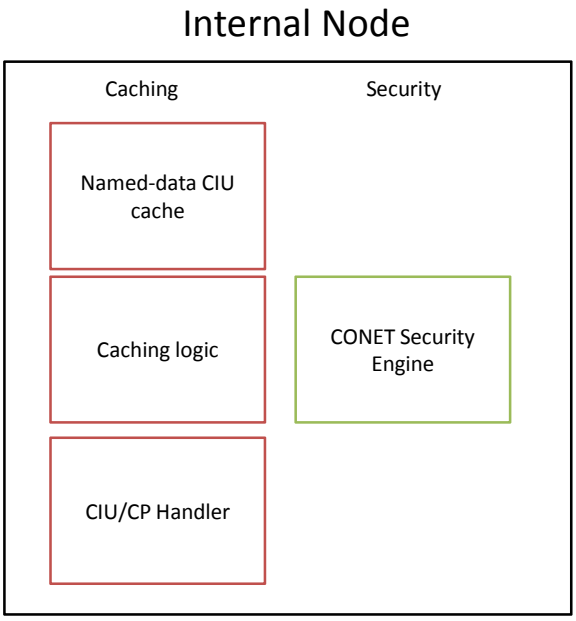


Figure 10 – Functional architecture of a internal node

Serving-Nodes

Figure 11 depicts the functional architecture of a serving node. A serving node provides forwarding, repository and security functionality. These functions work in the same way as on border-nodes. The only difference is that service nodes have a repository of named-data CIUs rather than a cache.

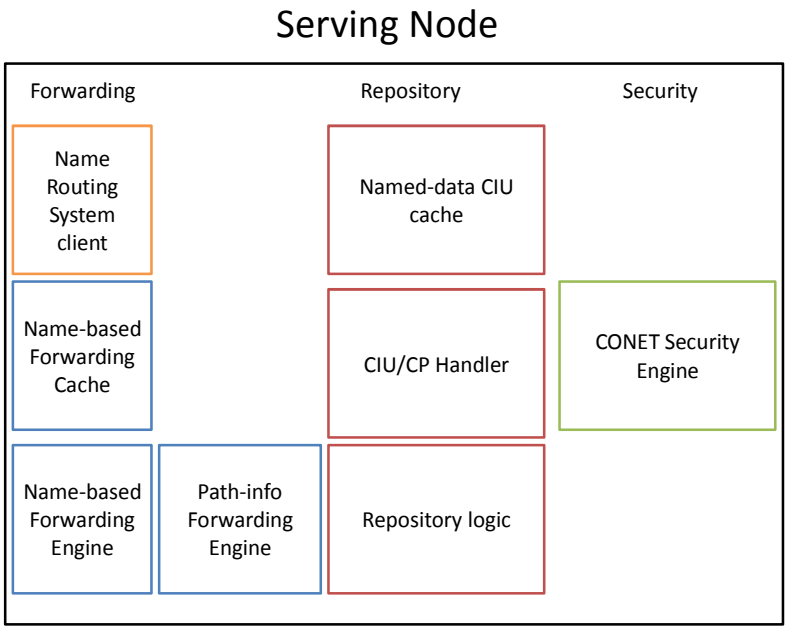


Figure 11 – Functional architecture of a serving node

The repository function stores the named-data items offered by the serving-node. The repository logic controls storage and delivery operations, using mechanisms similar to those used by the caching logic. After a new named-data item is stored in the serving-node, the Name Routing System Client function advertises the availability of the new data item to the Name Routing System.

End -Node

Figure 12 depicts the functional architecture of an end-node. Unlike border-nodes and serving-nodes, the end-nodes do not provide repository or caching functions. Another difference with respect to border-nodes and serving-nodes is that end-nodes have a transport function that controls the transport of whole named-data items. For reliable transfer, this function implements a receiver-driven TCP.

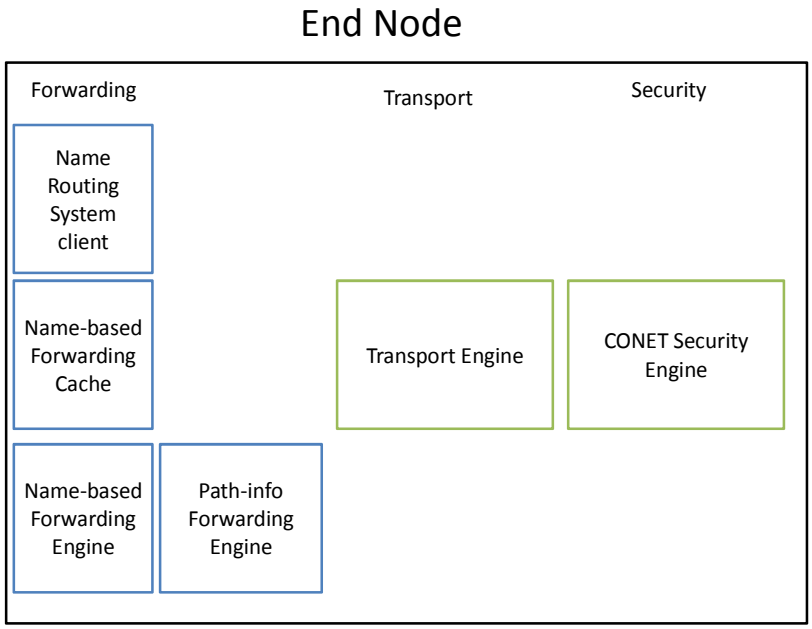


Figure 12 – Functional architecture of a end-node

Name Routing System node

Figure 13 depicts the functional architecture of a Name Routing System (NRS) node.

An NRS node has a database that contains all mappings between NIDs and Border Node CSS addresses required by a given CONET Sub System (CSS). The database is populated using a name-based routing protocol, which allows different NRS nodes to share reachability information for named-data items. An NRS server function allows NRS nodes to interact with other CONET nodes when looking up a forwarding entry or inserting a new forwarding entry.

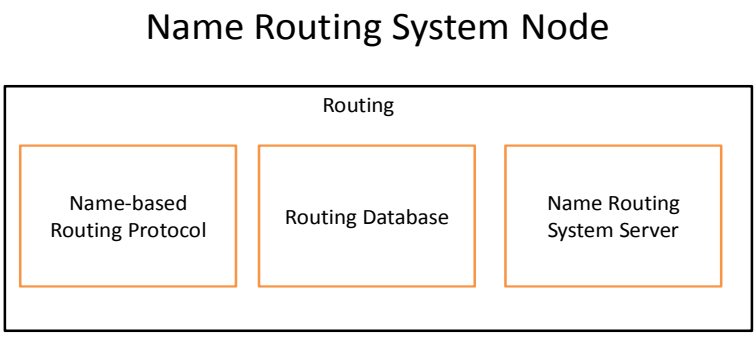


Figure 13 – Functional architecture of a Name Routing System Node

3.2 CoMid features

The following sections briefly introduce the key functionalities of CoMid. These functionalities allow calling applications to:

- Operate on VDIs, describing resources with metadata extracted from well-known or custom semantic taxonomies of concepts
- Publish information on resources into an overlay of peers whose topology is based on these same semantic taxonomies; this overlay will be referred to as Semantic Overlay
- Search for resources in specified regions of the Semantic Overlay and deliver matching content to users, even when resource descriptions do not fully match user requests.

3.2.1 The VDI

CONVERGENCE users act on Versatile Digital Items (VDIs). These are XML structures containing:

1. Identifiers
2. (links to) Resources
3. (links to) Semantically-rich metadata describing resources
4. (links to) Licenses expressing which rights are given to act on resources
5. Event report requests (ERR) instructing a peer to issue an event report (ER) to a specific user/peer in the event certain actions (e.g. play, store or match) are performed on a resource

So far, CONVERGENCE has defined 4 types of VDI: Resource (R-VDI), Publication (P-VDI), Subscription (S-VDI) and User (U-VDI).

As part of the creation process, a VDI is assigned a unique and persistent identifier. The CONVERGENCE Ontology Service allows users to establish links between VDIs, whenever a VDI is “superseded” by a new version or a new VDI is linked to an existing VDI.

3.2.2 Semantic and Dictionaries

The CONVERGENCE Core Ontology (CCO) is a native component of CONVERGENCE. The CCO allows for a semantic organization of peers in a virtual overlay network of "fractals", which are dynamically shaped and connected, on the basis of users' interests in different types of content. Peers join or leave a fractal based on what users are currently publishing and subscribing to. Figure 14 shows the basic structure of the CONVERGENCE Semantic Overlay. A peer typically belongs to more than one fractal, depending on how many users are interested in a specific kinds of content. To provide redundancy and mitigate peer churning (peers may be offline when users wish to access them), fractals are usually populated by more than one peer.

Domain and user ontologies can also be used to define fractals. Typically the former come from Service Providers (SP), while the latter are created by individual users. Fractals thus cover multiple "dimensions", each defined by an ontology. In this way, a single peer can reference resources described using different ontologies. Although fractals do not necessarily depend on ontologies - it is also possible to define a fractal using simple keywords – ontologies can make a major contribution to the discovery process. CONVERGENCE thus promotes the use of ontologies for all fractals.

The Community Dictionary Service (CDS) component is also part of the middleware and is implemented as a Technology Engine (CDS TE). This component maintains *dictionaries* that help translate concepts and properties from one ontology to another. The role of CDS is twofold. A user can use the CDS to describe her resources both when she creates R-VDI or a P-VDI and when she formulates a search query (while creating a S-VDI). In both cases, The CDS provides ontology entities leading to the creation of rich semantic metadata. The CDS can also facilitate the matching procedure by semantically expanding user descriptions and search queries, supporting thus non exact matches between PVDI descriptions and SVDI search queries.

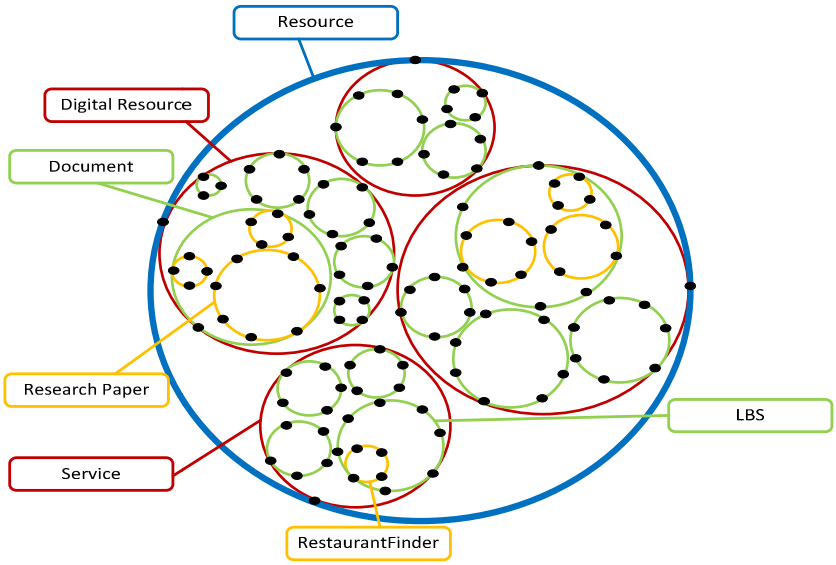


Figure 14: An example of CONVERGENCE fractals

3.2.3 Publish/Subscribe

A key interaction pattern for CONVERGENCE users is publish/subscribe. Users who make resources available to the system and discoverable are said to publish them. Users searching for specific resources are said to subscribe to them. The interaction is asynchronous and decoupled in time and space. Publications and subscriptions are both described by semantically rich metadata and labelled with relevant concepts. To find relevant content, a dedicated CoMid engine executes a semantic query, representing the subscription. The query is successful when published content matches the subscription criteria.

Fractals in the CONVERGENCE Semantic Overlay represent focused concepts, i.e. unions and/or intersection of concepts in the core ontology or other ontologies (dimensions). The use of "focused" fractals makes it possible to restrict the search space to peers actively participating in the fractal. All P-VDIs and S-VDIs labelled with a certain concept are stored in a specific set of peers in the fractal. Given that at any given time, some peers will be inactive, information is stored in multiple peers, providing redundancy.

Peers can:

1. Perform matches between P-VDIs and S-VDIs
2. Communicate any match to specified peers in the form of ER (Event Reports), depending on licenses and ERRs (Event Report Requests)
3. Remove S-VDIs and P-VDIs from the match tables when
    a. Their expiration date has passed
    b. An authorized user requests to remove them before the expiration date

4. Aggregate ERs from different peers and communicate the result to the end user

3.3 CoNet features

In this section, we briefly recap the main features of CoNet. Interested readers are urged to read section 7.1, which provides a comprehensive view of this important aspect of the CONVERGENCE system.

In CONVERGENCE, retrieval of resources and communications between peers is made possible by an Information-Centric Network (ICN), that we call CoNet. In contrast to the current Internet, in which users address remote hosts, CoNet allows them to access remote named-resources.

As shown in Figure 15, *named-resources* can consist of:

1. *Named-data*: i.e. a sequence of bits, like a VDI or the resource the VDI refers to
2. A named-service-access-point (*named-sap*): i.e. a network endpoint from which a Protocol Engine (or any other kind of service entity) receives CoMid messages.

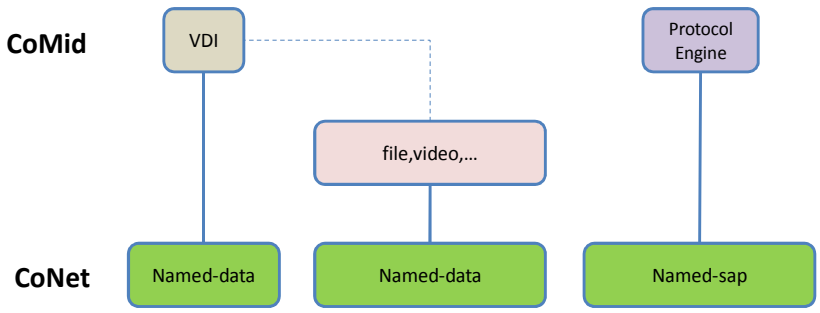


Figure 15: Mapping among CoNet named-resources and middleware entities

In both cases, the named-resource is identified by a *network-identifier*, i.e. a *name* like "foo:VDI1". In the current CONVERGENCE implementation, network identifiers coincide with the identifiers that a CoMid service assigns to VDIs and their components.

Note that the CoMid level and CoNet component in the Computing Platform level are completely decoupled: a VDI is a CoMid data-unit. For the CoNet, a VDI is simply a sequence of bits addressed by its network-identifier. The CoNet has its own data-unit, defined later in this report (see 7.1).

CoNet is aware of the network location of named-resources and uses *routing-by-name* to route the request to the copy of the named-resource held by the network node that is closest to the requesting user. The copy could be in a cache or in a serving-node (i.e. a server). Compared to traditional information-centric architectures (e.g. CCNx), CoNet limits the size of name-based routing tables by

including only a subset of all named-resources. Missing entries are looked up in a name-system and then cached. Unlike alternative network architectures, CoNet does not use the state of network nodes to deliver resources from a serving-node (or a cache) to a requesting peer. These features guarantee that CoNet will be more scalable with respect to the number of named-resources and number of ongoing communications.

CoNet supports *built-in caching/replication* functions: the same named-resource can be replicated in different network nodes, allowing users to access the most convenient replica. Replica nodes can be pre-provisioned, as in Content Delivery Networks, or opportunistically selected by *in-network caching* mechanisms. In-network caching reduces the risks associated with "flash crowds", i.e. situations when a very large number of users simultaneously access a popular resource. Unlike state-full and off-the-shelf proxy technologies, CoNet caching is stateless. This speeds up caching operations and reduces the cost of implementation.

CoNet provides *content-based quality of service*. Network nodes can offer different levels of bandwidth and storage (caching) according to the name of the resource they are serving. For instance, they can offer the named-data "foo:VDI-high-priority" a higher probability of being transmitted and a higher probability of being cached locally than the named-data "foo:VDI-low-priority". Unlike current IP technology, content-based QoS mechanisms do not require complex and slow deep packet inspection (DPI).

CoNet also provides *digital forgetting.* CoNet can remove a named-resource from all serving-nodes and caches on an explicit request from a resource owner, or when the resource has passed its expiry date.

3.4 CONVERGENCE Security and CoSec

3.4.1 Overview

Security is an essential feature of CONVERGENCE. The system's main security features are: i) assurance of VDI integrity (and authenticity); ii) governance of VDI access restrictions (confidentiality); iii) user identification and authentication; iv) issuing and enforcement of licenses; v) protection of user privacy; and vi) network security.

Most of these features are provided at the middleware level (CoMid), some at the Computing Platform level. CoMid security features are provided by the Security TE, which in turn relies on features offered by CoSec in the Computing Platform. Some security features of the CoNet are provided by the CoNet itself; others require support from the CoSec.

From a methodological standpoint, the CONVERGENCE project defines *security assets*, considers *threats* to these assets, and derives *security functional requirements,* which drive the selection and

development of appropriate cryptographic primitives and protocols. This methodology, which we apply to each of the issues just described, is described in greater detail in sections 4.15 and 4.16.

A key task of the methodology is to identify the "roles" that lie at the base of CONVERGENCE security. Note that this task is logically prior to the identification of the architectural entities (Technology Engines and Protocol Engines) that will implement the required functionality in the CONVERGENCE architecture (see section 3.1.2).

Two key roles are those of the *Identity Provider* (a trusted third party responsible for the *registration* of users, identification of users, verification of credentials, and issue of certificates) and the *Service Provider*, the entity that handles "daily" business in a specific scenario, including user authentication and licenses. One of the benefits of separating the two roles is *privacy protection*: Identity Providers never gain access to data accumulated by Service Providers; Service Providers do not need to know all (or any) of the personal data Identity Providers collect during registration. A typical *Identity Provider* might be a *government agency* that performs end-user registration and supplies Service Providers (e.g. *insurance companies*) with users' certified credentials.

3.4.2 CoMid Security

The CoMid exploits the features offered by the CoSec component in the Computing Platform through a dedicated Technology Engine called Security TE. Components wishing to perform a security function or protocol use CoSec via the Security TE. The CoSec serves all requests coming from the other engines and directed to the Security TE.

Based on CoSec, the Security TE can: i) create new credentials and manage certificates; ii) generate keys and encrypt/decrypt data or keys; iii) store confidential information (e.g. licenses and keys) in the secure repository; iv) certify the integrity of engines.

Other Engines rely on the Security Engine to perform: i) signing of VDIs (VDI TE); ii) symmetric encryption/decryption of resources (Media Framework TE); iii) asymmetric encryption/decryption of keys (REL TE); iv) user Identification (Identify User TE); v) user authentication (Authenticate User TE) (see Figure 16).

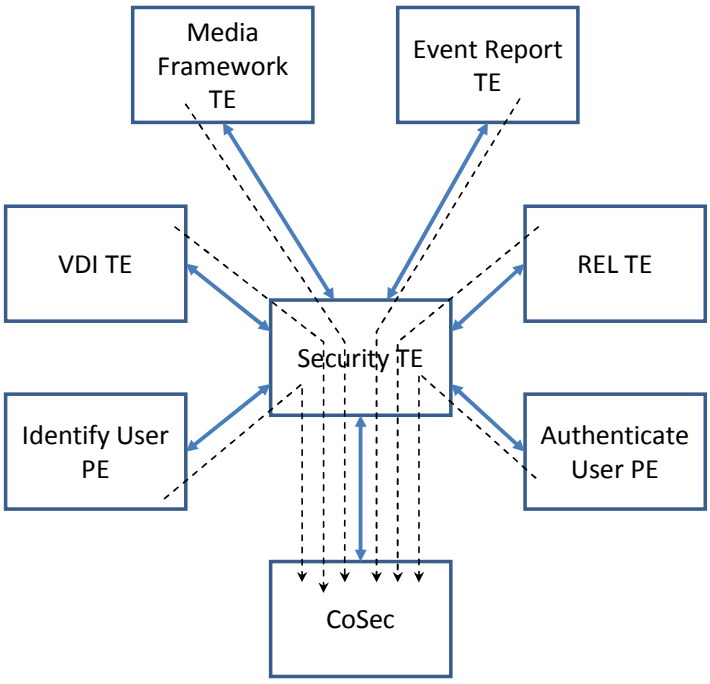


Figure 16: Security for CONVERGENCE CoMid

Secure repositories play a particularly important role in the CoSec security architecture. The preferred option for implementing secure repositories in CONVERGENCE will be a smart card. Unlike a piece of software, a smart card is highly tamper-resistant. Smart cards also have the ability to *process* security relevant protocols (e.g. issue and validate signatures, validate certificates, generate key pairs, etc.), providing functionalities that go far beyond secure storage.

3.4.3 The CoSec

The *CoSec* component in the Computing platform level is responsible for handling the majority of cryptographic protocols and security related tasks. Although the CONVERGENCE architecture diagram shows it as a single monolithic block, it actually has a *distributed architecture* encompassing several independent (and possibly distant) components, each of which includes software as well as hardware.

Most components of CoSec are located on *client computers* (e.g. end-user laptops), *smart cards*, *application servers* and *network peers*. The majority of protocols processed within CoSec involve *several* of these entities. The figure below illustrates the general style of these protocols as exemplified by a simplified user authentication protocol. In the example, the *smart card* acts as a *secure repository* for the user's signature private key.

CoSec features are exposed to the CoMid components (engines) through the CoMid Security TE. A CoSec API is defined so that the Security TE can use the CoSec. The CoNet can access CoSec features through the CoSec API.

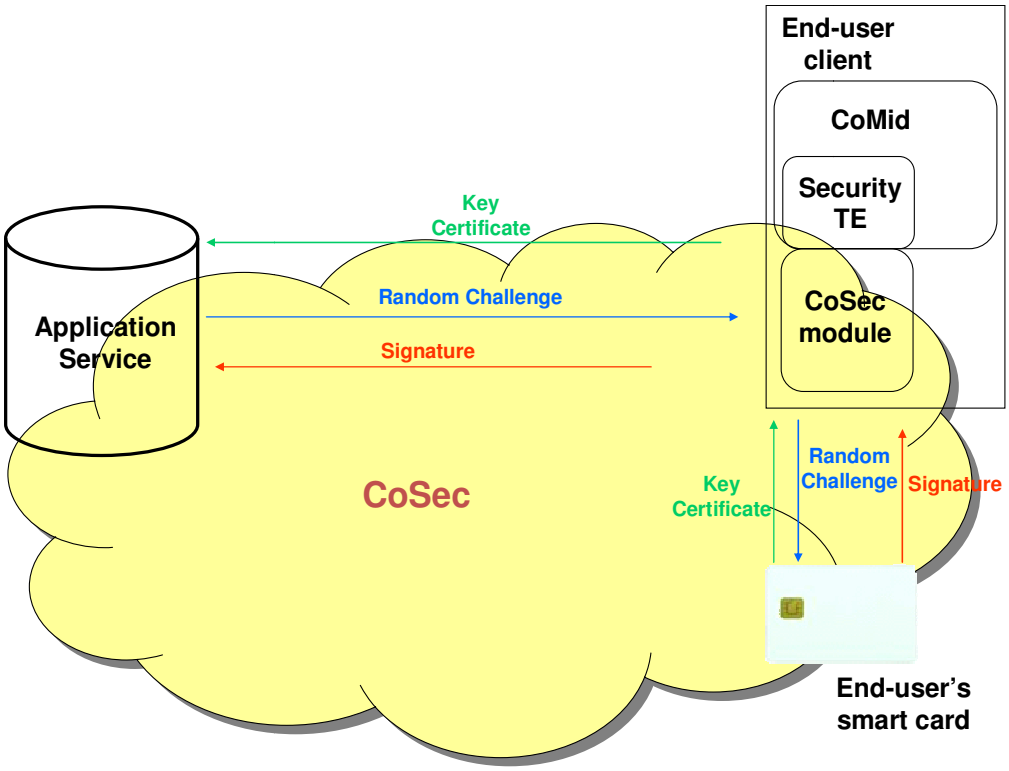


Figure 17 CoSec elements and example information exchange

3.4.4 CoNet security

CoNet supports security and privacy mechanisms aimed at preserving the integrity of the networking service and, where required, the anonymity of owners and consumers of named-resources. A distinguishing aspect of CoNet security is the use of *data-centric security*: security information is embedded in CoNet data-units. Data-centric security makes it possible for user and network nodes to verify the validity of named-resources, preventing the caching and dissemination of fake versions. Protecting information at the source (i.e. protecting the data unit) is more flexible and robust than delegating this function to applications, or securing only communications channels. CoNet security operations may be supported by CoSec.

3.4.5 Cryptographic primitives and protocols in CoSec

CONVERGENCE will use various cryptographic protocols to implement security requirements (see table below), including both off-the-shelf solutions and new primitives, which are still the object of ongoing research.

Fast *symmetric encryption* and *decryption* of content (partially off the shelf): this will use algorithms like AES-CBC; key escrow may be standard or based on sophisticated protocols such as ABE (Attribute Based Encryption), IBE (Identity Based Encryption)
Asymmetric cryptography (off the shelf): this will use established primitives like RSA or Elliptic Curves; asymmetric cryptography will be used for key agreements, signatures, certificates, key wrapping, etc.
Basic primitives like cryptographic hashes - (off the shelf)
Group Signature Protocol – (current research): group signature protocols will allow a member of a pre-specified group to anonymously sign a VDI's content or a challenge during authentication. It will only be possible to break anonymity on request (e.g. by a government entity with the appropriate legal powers)
Identity and Attribute Based Encryption - (current research): in this scheme, recipients of content (or messages) are assigned specific (arbitrary) attributes; the provider of content can encrypt according to these attributes, so that the possession of the same attributes is needed to decrypt the content.
Pseudonymous access via "Restricted Identification" - (current research) each user identifies herself with a pseudonym unique to a given context; the scheme does not allow linking of pseudonyms used in different contexts.

Section 7.2.2 will provide more details on cryptographic primitives and protocols.

4 Main technical and architectural concepts

4.1 The MPEG-M standard

The emerging MPEG-M standard (ISO/IEC 23006 – Multimedia Service Platform Technologies) is a suite of standards providing Architecture, Technology Engines (TE), Protocol Engines (PE), Aggregation Technologies and Reference Software that are well aligned to the basic concepts of CONVERGENCE, as represented in Figure 3 (Overview of CONVERGENCE architectural levels). The standard includes:

- Part 1 Architecture
- Part 2 MPEG Extensible Middleware
- Part 3 Reference Software
- Part 4 Elementary Services
- Part 5 Service Aggregation

The peer architecture described earlier in this document is based on MPEG-M. Part 2 introduces the notion of Engine, i.e. an appropriate groupings of technologies and the Application Programming Interfaces (API) through which an Application can access the functionalities it needs. As an Application typically needs more than one engine ("chains" of engines), MXM also provides examples of Orchestrator Engines, special MXM Engines capable of creating chains of Engines to execute, high-level application calls such as "Play" (see Figure 18).

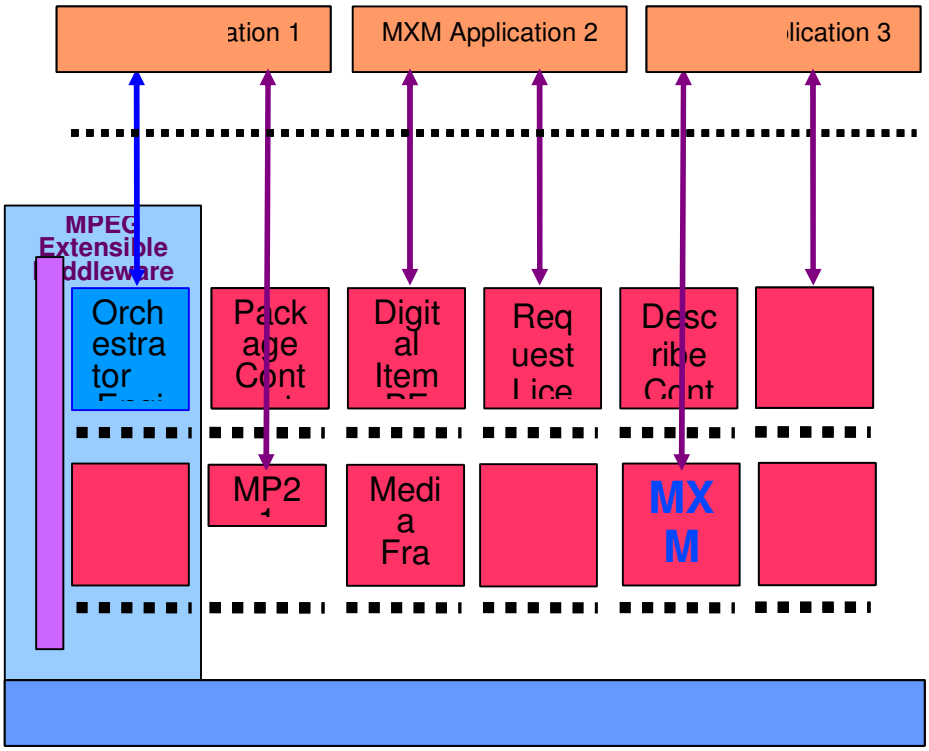


Figure 18 – The MXM standard

Part 4 introduces the notion of Elementary Services and specifies the messages exchanged between two peers. An implementation of protocols as specified by Part 4 is called a Protocol Engine. Finally Part 5 specifies how Protocol Engines can be chained to provide so-called Aggregated Services.

Recent years have seen the appearance of many new digital media related services, often involving combinations of Elementary Services. MPEG has shown that standardizing a set of technology elements and communication protocols facilitates the creation of aggregated services from a set of standard Elementary Services, even on demand.

Assuming that there is a Service Provider (SP) for each Elementary Service, a User may ask the Post Content SP to get a sequence of songs satisfying certain Content and User Descriptions. Figure 19 below depicts how a chain of Services could respond to such a user request.

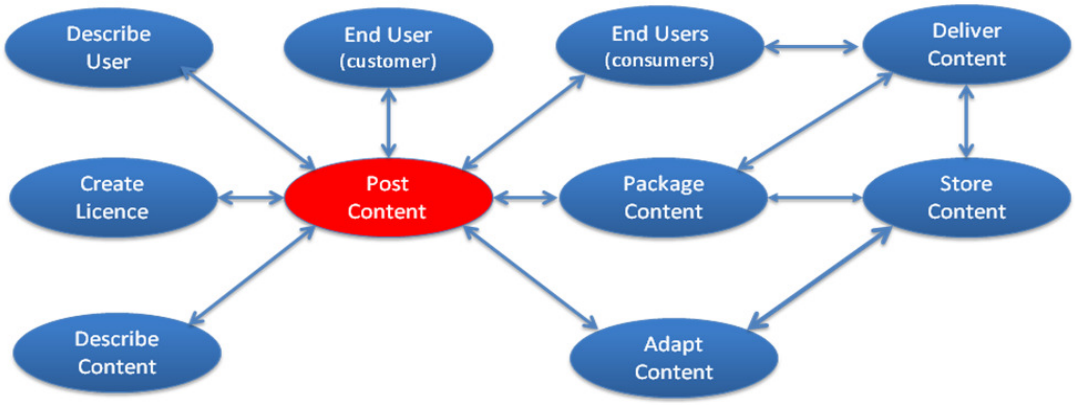


Figure 19– A possible services chain centred around Post Content SP

End User contacts Post Content SP who gets appropriate information from Describe Content SP and Describe User SP to prepare the sequence of songs using its internal logic. She then gets the necessary licenses from Create License SP. The sequence ("titles") of songs is handed over to Package Content SP. Package Content SP gets the songs ("Resources") from Store Content SP and hands over the Packaged Content to Deliver Content SP which streams the Packaged Content to End User.

MPEG has specified a set of standard Elementary Services and related protocols, enabling distributed applications to exchange information about entities that play a role in digital media services (e.g. Content, Contract, Device, Event and User), and the processing that a party may wish to execute on those entities, (e.g. Authenticate, Create, Deliver, Describe, Identify, Negotiate, Process, Request, Search and Transact). These have been standardized in part 4 "Elementary Services" [31].

Given these advantages, CONVERGENCE has decided to adopt MPEG-M, the MPEG-M protocol , MPEG Technology Engines, and MPEG-M aggregation technologies as the basis of its middleware architecture. Many CONVERGENCE Protocol and Technology Engines are taken directly from MPEG-M.

CONVEGENCE also extends the MPEG-M standard. In particular, it extends the Describe Content PE to support integration with ontologies (CDS).

Specific CONVERGENCE functionalities are supported by other new engines:

Protocol Engines
1. Describe Content (extended functionality)
2. Inject Content (new)

Technology Engines
1. Community Dictionary Service (CDS)
2. CoNet
3. Match
4. Overlay
5. Security (extended)

CONVERGENCE is developing and testing its new engines, and plans to propose some of them for standardization.

4.2 Information Centric Networking

Several recent papers (e.g. [11][15][7][16]) and research projects ([8][9]) have proposed a shift from "host-centric networking" to "Information-Centric" or "Content-Centric" networking. The essence of Information-Centric Networking (ICN) is that the network layer provides users with content, instead of communication channels between hosts, and is aware of the content, at least in the sense of knowing its "name". A ICN architecture should:

- Address contents, using an addressing scheme based on names, which do not include references to their location
- Route a user request, which includes a content-name, toward the closest copy of the content with such a name (name-based, anycast routing) and deliver the content to the requesting host
- Provide native, in-network caching functionality to achieve efficient content delivery in both fixed and mobile environments [17]
- Exploit security information embedded in the content to avoid the diffusion of fake versions of content and to protect the content (a more robust solution than entrusting security to applications, or securing only the communications channels [16])
- Provide a way to differentiate the perceived quality provided by different services [18], and provide per-content quality of service differentiation, covering cache hits.

Network level functionalities in CONVERGENCE are provided by the CONVERGENCE Network (CoNet). CoNet is an Information-Centric inter-network that provides users with network access to remote named-resources over a public or private network infrastructure .

In the CONVERGENCE CoNet, named-resources can be either data[3] ("named-data") or service-access-points[4] ("named-service-access-points"), identified by a network-identifier (a name).

As shown in Figure 20, CoNet interconnects CoNet Sub Systems, which can be layer-2 networks, layer-3 networks or couples of nodes connected by a point-to-point link. CoNet supports both the "clean-slate" and "overlay" approaches to deployment, proposed in the literature. In addition, CoNet supports a novel "integration" approach, which extends the IP layer with a new header option that makes IP content-aware[6].

CoNet limits the size of name-based routing tables by including only a subset of all named-resources; missing entries are looked up in a name-system and then cached. CoNet does not maintain states in network nodes during delivery of content.

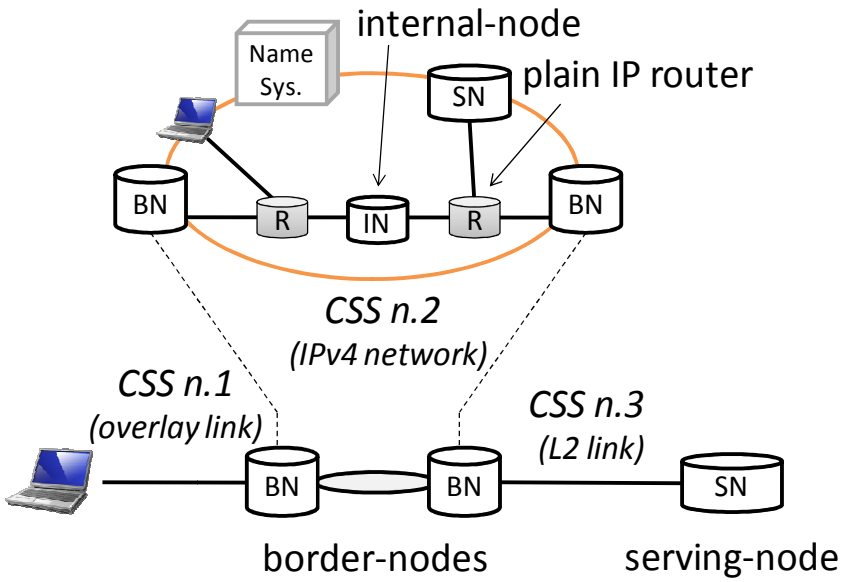


Figure 20: CoNet Architecture

[3]Named-Data includes: documents, video, images, structured information, VDIs. The Network level is general purpose, in the sense that it can handle any kind of data and not only data generated by the CONVERGENCE system.

[4] A named-service-access-point is a network endpoint through which an upper layer entity (e.g., a server or a client) sends and receives data. In the current Internet, for instance, TCP port n. 80 is the default service-access-point for HTTP servers.

4.3 VDI

The basic design of CONVERGENCE makes a functional distinction between different kinds of VDIs.

CONVERGENCE's basic building block is a **Resource** – the bits representing a multimedia stream or the digital representation of a real-world object, for instance.

As a first conceptual step, we create a **Resource VDI** around the resource, and let users store it somewhere in the network.

The second conceptual step is to publish the resource to the world. To achieve this, the system uses a generic gossiping protocol over an overlay of peers (the CONVERGENCE Semantic Overlay). The information to be gossiped is gathered from the Resource VDI, packaged into a transport package and gossiped from peer to peer. The transport package is also a VDI. We call it a **Publication VDI**.

The reason we use VDIs to advertise the publication of another VDI is because it is convenient: all the information that needs to be gossiped (metadata, license, event reports) fits naturally in a VDI and the gossiping protocol does not care about what it is transporting. However, the most important reason is that when information about a publication is packaged as a VDI, it can be stored in the physical network in the same format, independent of the overlay. By observing the flow of VDIs across the network, external entities that know the VDI schema can collect statistics about what is being published and subscribed to on the system.

Publication VDIs offer an additional level of indirection between the Resource and the Resource being advertised, an important advantage in complex business scenarios. Publication VDIs allow users to publish the same resource, with different metadata, at different times, using different identities. For instance, a "second publisher" with appropriate permission from the author, can also publish the resource – perhaps because she wants to publish it in a different context. All she needs to do is provide a new Publication VDI, pointing to the same Resource VDI. The CONVERGENCE system tries not to restrict users' ability to invent new patterns of business so long as they hold the necessary permissions. Publication VDIs carry a license and an expiry date.

Subscription VDIs, issued by users searching for specific resources, play a role symmetrical to that of Publication VDIs and also carry a license and an expiry date.

4.3.1.1 Independence of Publication VDIs from Resource VDIs

The introduction of Publication and Subscription (Pub/Sub) VDIs decouples the functionalities of the middleware and the network level. CoMid is all about matching publications and subscriptions, and discovery, and basically does not deal with content. To use an analogy with today's Internet, it is the equivalent of the Google database. CoNet, by contrast, is equivalent to ISP servers that store web pages. The two levels are independently managed. This decoupling makes the system more robust.

Since the CoMid and CoNet are decoupled, the only point in common between a Resource VDI and its associated Publication VDI is the Resource VDI identifier, embedded in the Publication, which links them by reference. This is because they serve two completely different purposes. A Resource VDI represents a "thing"; a Publication VDI represents information about the "thing".

In principle, a Resource VDI and its associated Publication VDI can have distinct expiry dates. The expiry date of the Publication VDI determines how long the resource will be discoverable. When it expires, the Resource VDI will no longer show up in searches and subscriptions (it will no longer be visible at the CoMid level). However, it will still be possible to fetch it using a direct link held in the CoNet level. In many practical situations, the Resource VDI will not have an expiry date. In other cases, the Resource VDI may expire before the Publication VDI referring to it. This makes it possible to conserve information about a resource, even when the resource itself is no longer available. Resource VDIs can circulate in the system without being associated with a Publication VDI. This means they can be indexed by crawlers but are not part of the pub/sub process. Expiry of such resources is managed by CoNet expiry mechanisms, which are independent of mechanisms in CoMid and work independently of whether they have been published.

CONVERGENCE is designed to facilitate automated generation of Publication VDIs from Resource VDIs. If publisher and author are the same person (if the principal in the two Licenses is the same) the process can be fully automated. In the default scenario, the two VDIs are given the same expiry date, though users can configure different expiry dates if they wish. If publisher and author are different persons (the system can tell), the Publish Aggregated Service will ask the publisher whether she wishes to keep the licensing conditions defined in the Resource VDI or define new ones.

4.3.1.2 Storage of Publication and Subscription VDIs

Storing Publication and Subscription VDIs on the network has important advantages: the format of the database that indexes resources is well-known, and the database itself is highly available (though visibility of some VDIs may be restricted by License).

Possible strategies for assigning Publication and Subscription VDIs to networks nodes include storing them in the same peer where they have been created, or on the first peer they are gossiped to (providing the peers deploy the full CONVERGNCE stack, down to the CoNet level). The project has not yet taken a final decision on this issue. However, it is a clear requirement that VDIs should always be reachable from CoNet, using well-specified names.

4.3.1.3 User VDIs

The project is exploring the possible introduction of another kind of VDI, which we call the **User VDI**. This allows for a level of indirection between CoMid services and the real location and/or identity of the end-user accessing services. This concept will be further explored and refined in future deliverables.

4.3.1.4 The Unpublish and Unsubscribe VDIs

The project is currently studying the definition of UnPublish and UnSubscribe VDIs. Given that peers will join and leave the system (as they go on or off line), in an arbitrary fashion, it is not possible to disseminate information about the revocation of a Publication or Subscription VDI to all peers in a synchronous and fail proof way. Whatever information spreads across the system at a specific instant, there will always be peers that miss it, because they are off-line.

To deal with this issue, Unpublish and Unsubscribe commands must be distributed asynchronously. The injection of UnPublish and UnSubscribe VDIs into the system signals a specific command (unpublish or unsubscribe), which is asynchronously gossiped as peers come online and go offline. This way, the information is disseminated to all peers as soon as possible.

4.4 VDI naming scheme

Ideally a naming scheme should be

- **Secure**: the name of the resource is unique and guaranteed to identify the correct resource
- **Decentralized**: there is no central authority that controls all names
- **Human-readable**: the name is human-readable (possibly with a meaning) and is memorable.

However, as Zooko's Triangle specifies, no single naming-scheme can meet more than two of these requirements at the same time (see figure below).

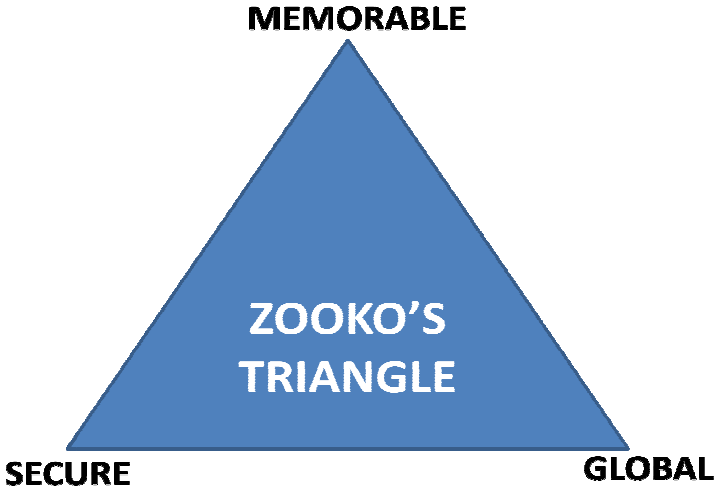


Figure 21- Zooko's Triangle

CONVERGENCE will thus adopt a flexible naming scheme. One proposal, described in section 7.1.3, is to split the name in two parts:

- a *namespace/principal* part, which is secure and human readable

- a *name* part, which is decentralized and human-readable

When combined, the two parts offer a name that is completely human-readable, a secure (centrally managed) principal, and a locally managed name.

4.5 VDI dynamics and linking

Current systems for versioning of on-line digital content and for creating links between content rely on rigid, semantically mute or ambiguous connections between digital objects. More specifically, the only standard and universal (HTML based) way of declaring a connection between two digital documents is to insert the URL for one of them in the other document. The usual way of doing this is to use *<a>* or *<link>* tags, indicating that the referenced document is somehow related to the referencing one. The specification of the tags provides some support for the expression of the nature/type of these relationships (3.4.2.1 [20]). Nonetheless, the number of types that can be expressed is very low and the types themselves are semantically ambiguous. Hardly any of the major on-line content retrieval and access systems (browsers) use the information.

CONVERGENCE aims to change this situation by providing users with a universal, explicit, rich and semantically unambiguous way of declaring and exploiting relationships between published digital objects (VDIs). In this way, CONVERGENCE will provide interesting new features.

- Clear, universal and unambiguous declarations of versioning relationships between VDIs will make it possible to group them into sequences of subsequent VDI versions and provide a universal and explicit mechanism for updating digital documents.

- Universal and unambiguous declarations of a base set of logical relationships between VDIs will enable users to declare a VDI to be a comment or correction to another VDI, and allow the system to automatically and unambiguously detect and process such relationships.

- Context dependent declarations of logical relationships between VDIs (or between sections of VDIs) will allow users to define “application scope” or “VDI scope” relationships, which can be interpreted using a domain-specific ontology.

4.5.1 Functional Requirements

CONVERGENCE supports two main mechanisms for expressing the relationship between VDIs.

- The first mechanism consists of a declaration that a VDI belongs to sequence of VDIs in which each VDI in the sequence is derived from the previous node in the sequence. In this way the first node in the sequence is the initial VDI and the last node is the most recent one.

- In the second mechanism, every VDI contains a declaration, inside the VDI, of its logical relationships to other VDIs. For example, if VDI B is a correction to VDI A, VDI B will contain an explicit declaration that VDI B maintains a relationship of "*correction*" to VDI A.

The implementation of these mechanisms is described in D4.1, sections 5.2, 5.3 and 6.3.

4.6 CDS

The Community Dictionary Service (CDS) is the CoMid component responsible for supporting CONVERGENCE semantic functionalities. To support this role, the CDS maintains and exploits ontologies and the relationships between them. CDS is a distributed service: each peer has access to a local CDS which can be customized with user ontologies, as well as remote access to richer CDSs.

The following sections briefly describe how the CDS supports semantic descriptions of VDIs, and the way these descriptions are exploited to provide semantic matching between publications and subscriptions.

4.6.1 CDS supports content description

Given Convergence's Information-Centric publish/subscribe paradigm (see section 4.11), it is vital to correctly match published VDIs to user subscriptions. This requires coherent semantic descriptions of VDI resources and of user subscription criteria. To define these criteria, users employ CDS servers, which contain ontology models that can accurately describe the resource they wish to publish and to subscribe to. Each user can run her own (local or remote) CDS server, loaded with custom/user ontologies as well as recognized domain ontologies. The following paragraphs describe how the CDS supports this process.

4.6.1.1 CDS involvement in publishing

To help users in building semantic descriptions of their VDIs, the CDS exposes a service for ontology entity exploration. The service allows users to select ontology entities to semantically describe their VDIs. The walkthrough below describes the creation of the semantic description of a VDI for the movie "Star Wars". The user's CDS is loaded with the Movie Ontology (www.movieontology.org) and the IMDB ontology [40]. The identifier of the VDI is set to RVDI_23.

Step	User action	System response	Semantic description
1	User types "movie"	CDS searches for an entity named movie and returns the imdb:Movie and the movieOntology:Movie Classes	
2	User selects movieOntology:Movie	System selects the movieOntology ontology for the	<rdf:rdf> <rdf:description rdf:about="RVDI_23"> <rdf:type

		description of the VDI and returns the rdf description	rdf:resource="&movieOntology;Movie"/> </rdf:description> </rdf:rdf>
3	User types title	CDS returns with the movieOntology:title DatatypeProperty	
4	User selects movieOntology:title	System prompts user to enter the title of the movie	
5	User types "Star Wars"	System returns the user the rdf description	<rdf:rdf> <rdf:description rdf:about="RVDI_23"> <rdf:type rdf:resource="&movieOntology;Movie"/> <movieOntology:title>Star Wars</movieOntology:title> </rdf:description> </rdf:rdf>

4.6.1.2 CDS involvement in subscribing

The same service can help users to define subscription criteria. The walkthrough below presents the creation of a SPARQL query for the movie "Star Wars". User's CDS is again loaded with the Movie Ontology and the IMDB ontology.

Step	User action	System response	Semantic query
1	User types movie	CDS searches for an entity named movie and returns the imdb:Movie and the movieOntology:Movie Classes	

2	User selects imdb:Movie	System selects the imdb ontology for the description of the VDI and returns the sparql query	SELECT ?x
			WHERE
			{
			?x rdf:type imdb:Movie
			}

3	User types title	CDS returns with the imdb:title DatatypeProperty	
4	User selects imdb:title	System prompts user to enter the title of the movie	

5	User types “Star Wars”	System returns the user the SPARQL query	SELECT ?x
			WHERE
			{
			?x rdf:type imdb:Movie
			?x imdb:title “Star Wars”
			}

4.6.1.3 Matching

Matching is one of the core issues that the CONVERGENCE system has to face. The plethora of ontology models and the freedom granted to users to select the ontology with which they describe their VDIs (or to use their own ontologies) inevitably creates a huge diversity in the metadata used to described VDIs. This creates a risk that user publications and subscriptions, described using different ontologies, will never match.

4.6.2 CDS dictionaries

4.6.2.1 Introduction

CDS dictionaries are a new concept designed to address the issues of heterogeneous metadata. In essence, a CDS dictionary is an ontology, just like user ontologies and domain ontologies. Unlike other ontologies, however, its whole content consists of a mapping between two different ontologies: that is, a set of equivalence statements between entities in the first ontology and entities in the second. In this way, CDS dictionaries provide semantic bridges between user-ontologies, between user-ontologies and domain ontologies and between domain ontologies.

The figure below depicts the set of ontologies the CDS uses during publication, subscription and matching procedures.

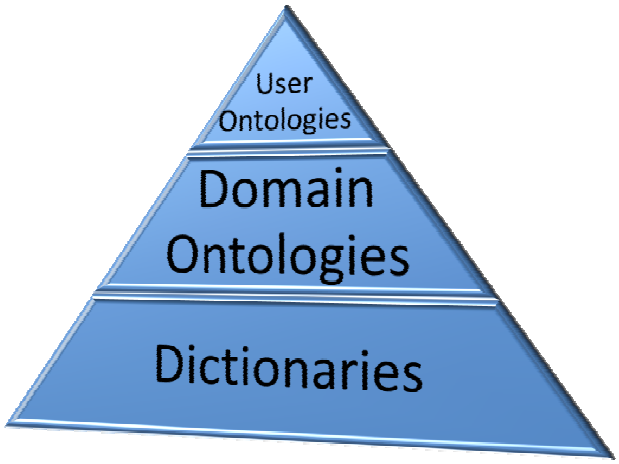


Figure 22 - CDS Ontologies

4.6.2.2 CDS dictionary format

As an example of the CDS dictionary format, the table below provides a mapping between the movieOntology and the imdb ontologies.

movieOntology ontology	`<owl:Class rdf:about="&movieOntology;Movie"/>`
	
	`<owl:DatatypeProperty rdf:about="&movieontology;title">`
	`<rdfs:domain rdf:resource="&movieOntology;Movie"/>`
	`<rdfs:range rdf:resource="&xsd;string"/>`
	`</owl:DatatypeProperty>`
	…

movieOntology-imdb dictionary	`<owl:Class rdf:about="&movieOntology;Movie">`
	`<owl:equivalentClass rdf:resource=”&imdb;Movie”/>`
	`</owl:Class>`
	
	`<owl:DatatypeProperty rdf:about="&movieontology;title">`
	`<owl:equivalentProperty rdf:resource=”&imdb;title”/>`
	`</owl:DatatypeProperty>`
	…

Imdb ontology	`<owl:Class rdf:about="&imdb;Movie"/>`
	
	`<owl:DatatypeProperty rdf:about="&imdb;title">`
	`<rdfs:domain rdf:resource="&imdb;Movie"/>`
	`<rdfs:range rdf:resource="&xsd;string"/>`
	`</owl:DatatypeProperty>`
	…

Dictionaries can also use the skos:narrower and skos:broader properties to connect ontology entities that are not semantically equivalent.

4.6.3 Mechanisms for exploiting CDS dictionaries

4.6.3.1 Publishing

Consider the "Star Wars" Resource VDI, described in the previous example. The metadata in the VDI is based on the movieOntology. The walkthrough below shows how the user can use the CDS to create semantically equivalent metadata based on the imdb ontology.

Step	User action	System response	Semantic description
1	User passes the movieOntology-based semantic description to the CDS		<rdf:rdf> <rdf:description rdf:about="VDI1"> <rdf:type rdf:resource="&movieOntology;Movie"/> <movieOntology:title>Star Wars</movieOntology:title> </rdf:description> </rdf:rdf>
2		CDS returns the user the imdb-based equivalent rdf description	<rdf:rdf> <rdf:description rdf:about="VDI1"> <rdf:type rdf:resource="&imdb;Movie"/> <imdb:title>Star Wars</imdb:title> </rdf:description> </rdf:rdf>

4.6.3.2 Subscribing

The CDS can also be used to translate a SPARQL query from one ontology to another, as shown in the walkthrough below.

Step	User action	System response	Semantic query

1	User passes the imdb-based SPARQL query to the CDS		SELECT ?x
			WHERE
			{
			?x rdf:type imdb:Movie
			?x imdb:title "Star Wars"
			}

		CDS returns the user the movieOntology-based equivalent SPARQL	SELECT ?x WHERE {

		query	?x rdf:type movieOntology:Movie ?x movieOntology:title “Star Wars” }

4.6.3.3 Matching

On receiving the publication, the CDS expands the semantic description of the VDI of the movie using the movieOntology-imdb dictionary. As seen in the walkthrough below, this procedure creates new triples.

Step	User action	System response	Semantic description
1	User passes the movieOntology-based semantic description to the CDS for materialization		<rdf:rdf> <rdf:description rdf:about=”VDI1”> </rdf:description> </rdf:rdf>
2		CDS returns the user the materialized rdf description of the VDI	<rdf:rdf> <rdf:description rdf:about=”VDI1”> <rdf:type rdf:resource=”&imdb;Movie”/> <imdb:title>Star Wars</imdb:title> <rdf:type rdf:resource=”&movieOntology;Movie”/> <movieOntology:title>Star Wars</movieOntology:title> </rdf:description> </rdf:rdf>

After expansion, movieOntology-based and the imdb-based SPARQL queries will both find the match. This mechanism is explained in detail in section 4.14.

4.7 Semantic Overlay

4.7.1 Semantic Foundations

CONVERGENCE is built upon an Information-Centric network, which is accessed through the CoNet engine and its APIs. Hence, applications are not aware of the locations where resources are stored. To

access a resource, they use the name under which it is advertised in CoNet. However, a name cannot describe the full content of a resource. This requires semantic metadata. The indexing and efficient retrieval of content using semantic metadata is the key feature of the CONVERGENCE **Semantic Overlay**.

Efficient semantic operations on metadata require a scheme for semantic categorization of resources [38]. The first element in the CONVERGENCE scheme is the so-called CONVERGENCE Core Ontology (CCO); a non-mandatory CONVERGENCE-wide hierarchical taxonomy of resource semantic types, implemented as an OWL ontology. The first level root concept in the ontology is the "Resource"; first level children are "Digital Resources", "People", "Real World Objects" and "Services", corresponding respectively to the *Internet of Media,* the *Internet of People,* the *Internet of Things and* the *Internet of Services*. Figure 23 provides an abstract view of the CCO.

Given that the CCO is non-mandatory, content can also be categorized using other ontologies, or even non-semantic technologies.

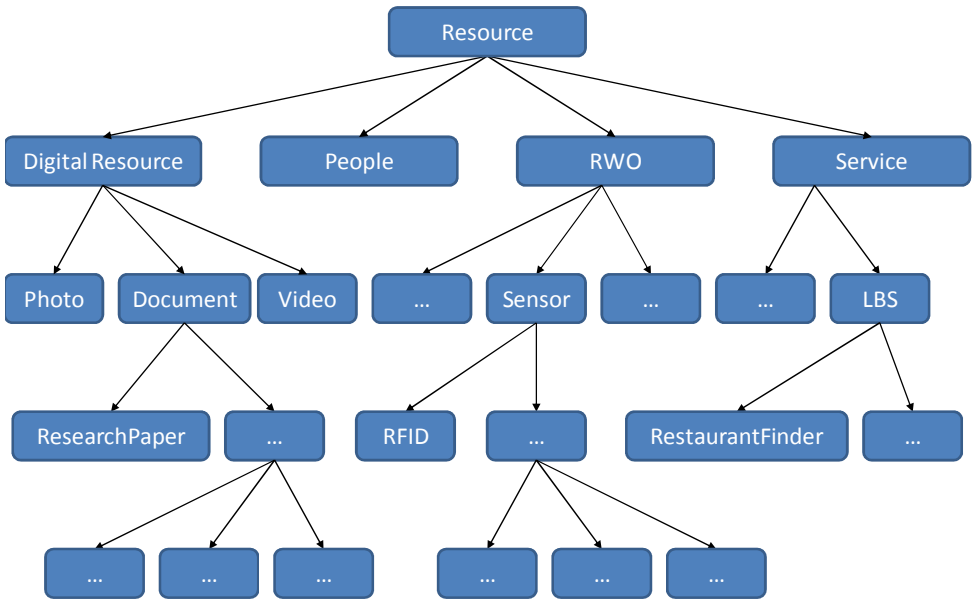


Figure 23 - CONVERGENCE Core Ontology (CCO)

The CCO is not the only ontology available to CONVERGENCE users. Additional ontologies make it possible to categorize content in other ways (e.g. specialist ontologies of music or movie genres, a classification of peers based on their physical location).

4.7.2 Semantically managing the Overlay Topology

Peers belonging to the overlay are partitioned into groups (usually) based on semantic criteria. For example, all peers which belong to the category cco:movie, belong to the group cco:movie. Groups like this have the same kind of self-similarity we find in fractals: they "can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole" [33]. We therefore call them *network fractals*.

A peer may belong to some fractals in the CCO and other fractals in another ontology – representing another *dimension* of semantic meaning (e.g. an ontology of access rights). This organization is *virtual* and not physical. Each peer belongs to one or more fractals, in one or more dimensions. Every time a peer decides to make a publication or a subscription, it determines the final group of peers that should receive the message, based on the metadata used to describe the publication or the subscription. These peers may belong to one or more fractals in one or more dimensions.

Consider a fractal *i* belonging to the *j*-th dimension and consisting of a group of peers, G_i^j. The message is then propagated to the peers of the group *G* that satisfy the condition:

$$G = \bigcap_{j} \bigcup_{i} G_i^j$$

This implies that:

- When a user wants to inject content into a set of fractals belonging to a single dimension, this content is available within the union of these fractals
- When a user wants to inject content into a set of fractals belonging to multiple dimensions, she takes the union of fractals for each dimension and injects the content into the intersection of these sets.

4.7.3 Propagation Protocol

A major challenge facing the CONVERGENCE Semantic Overlay is the very dynamic nature of the CONVERGENCE system with peers continuously joining and leaving fractals: for instance,

- A peer may no longer be interested in a topic (i.e. no publications or subscriptions for that topic are present in the peer) and leaves the fractal
- A new peer is interested in a topic (makes a publication and/or subscription for that topic) and joins the fractal
- A peer goes offline/online

- A peer crashes.

CONVERGENCE therefore needs a protocol to efficiently propagate content across the network and to enable peers to maintain a consistent albeit partial view of the network. This means that the protocol must ensure:

- Robustness: the protocols should ensure reachability up to significant churn rates (rate of peers leaving the network or new peers moving in)
- Scalability: the protocol should not require a peer to communicate with large numbers of other peers. Ideally networks should have a "small-world" configuration: the typical distance L between two randomly chosen nodes (the number of steps required) should grow proportionally to the logarithm of the number of nodes N in the network (O(logN)).

Ensuring robustness in very large small-world networks requires a state-of-the-art design.

The goal of the protocol is thus to maintain the topology of the overlay in such a way that the distance between peers that belong together is continuously reduced without compromising their ability to reach far away portions of the network. In practice, this means that the protocol has to continuously update each peer's list of connections to other peers in such a way that it moves progressively closer to its natural position in the network. In other words, peers have to continuously exchange profiles with the neighbours, and update their list of neighbours according to the profiles they acquire.

CONVERGENCE is therefore investigating state of the art technologies based on gossiping and distributed hash tables (DHT), to propagate content in the fractal and to enable peers to maintain a partial view of their fractals, since such protocols have been proven to be robust and scalable and, hence, a good solution for dynamic networks [34][35][37][36][43][44]. We are investigating a protocol that will combine the benefits of gossiping and DHT building a 2 level overlay architecture. The final decisions regarding the design of the protocol and the selection of the corresponding gossiping and DHT inspired techniques will be based on the evaluation of simulation results and illustrated in the forthcoming deliverables of WP5.

Another important challenge facing CONVERGENCE is fractal discovery. This issue will be discussed in deliverable D5.2.

4.8 Event reporting

Event Reporting provides a standardized mechanism for defining, identifying and sending notifications about events, where an event is defined by a set of conditions which, when met, triggers an event catch. The main conditions involve Users, VDIs and specific type of action executed on VDIs. Major classes of "reportable" Events include:

(a) Events specific to the publish subscribe process, triggered by a "Match" action;

(b) Events generated within a peer during processing of a VDI by an application (e.g. play of a Movie VDI, the creation of an annotation for a VDI, etc.).

The events falling under (a) occur when a newly published VDI matches user subscription criteria, generating an Event Report. The events falling under (b) are associated with the domain logic of a particular application and are specific to the application. This implies that before using Event Reporting, applications have to define and catch relevant events.

The CONVERGENCE Event Reporting mechanism involves two constructs.

1. Event Report Request (ERR). An ERR defines the conditions that trigger an event. Events defined by an ERR trigger the creation of an ER which contains information describing the event, as specified in the associated ERR.

2. Event Reports (ER). ER are generated and sent according to the ERR. When the event occurs, an Event Report is generated and delivered to the specified recipient(s). The report contains the information requested.

4.9 Rights Expression Language

A Rights Expression Language (REL) is a machine-readable language that can declare rights and permissions using terms with an agreed semantics. A REL provides flexible, interoperable mechanisms to support publishing, distribution, and consumption of digital resources. This ensures that rights, conditions, and fees specified for digital resources are honoured and that personal data are processed in accordance with individual rights.

The REL standardized by MPEG [25] adopts a simple, extensible data model for many of its key concepts and elements (see Figure 24). The MPEG REL data model for a rights expression consists of four basic entities and the relationship among those entities. This basic relationship is defined by the MPEG REL assertion “grant”. Structurally, an MPEG REL grant consists of:

1. The **principal** to whom the grant is issued
2. The **right** that the grant specifies
3. The **resource** to which the right in the grant applies
4. The **condition** that must be met before the right can be exercised

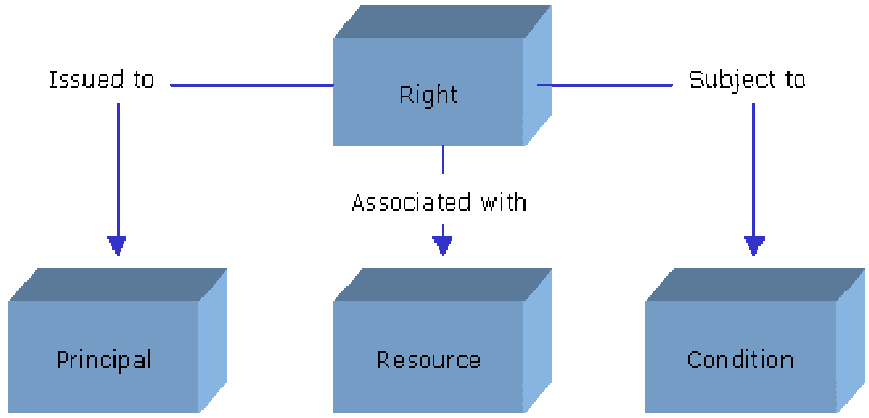


Figure 24 - The REL Data Model

Principal: encapsulates the identification of principals to whom rights are granted. A principal denotes the party that it identifies by information unique to that party. Usefully, this is information that has some associated authentication mechanism by which the principal can prove its identity.

Right: the "verb" that a principal can be granted to exercise against some resource under some condition. Typically, a right specifies an action (or activity) or a class of actions that a principal may perform on or using the associated resource.

Resource: the "object" to which a principal can be granted a right. A resource can be a digital work (such as an e-book, an audio or video file, or an image), a service (such as an email service, or B2B transaction service), or even a piece of information that can be owned by a principal (such as a name or an email address).

Condition: the terms, conditions and obligations under which rights can be exercised. A simple condition is a time interval within which a right can be exercised. A slightly more complicated condition is to require the existence of a valid, prerequisite right that has been issued to some principal. Using this mechanism, the eligibility to exercise a right can become dependent on the eligibility to exercise other rights. In Phase 1 of the CONVERGENCE trials, the project will use the existing language and semantics. In the subsequent phases, it will substantially extend the elements of the language to handle specific CONVERGENCE requirements.

4.10 Content identification

The CONVERGENCE system adopts three kinds of identifier: **VDI Identifiers, VDI Sequence Identifiers** and **Network Identifiers**.

- VDI Identifiers are used by CONVERGENCE Middleware (CoMid) to uniquely identify VDIs.

- The VDI Sequence Identifier is used by the CONVERGENCE Middleware (CoMid) to uniquely identify a sequence of VDIs so as to support information updating. The VDI Sequence Identifier is contained in a VDI.

When the information of a VDI needs to be updated, a new VDI is created with a different VDI Identifier, but with the same VDI Sequence Identifier. The VDI Sequence Identifier makes it possible to search for and retrieve the latest version of a VDI, or any preceding version. A VDI Sequence identifier has the form of a URI as better specified in D4.1 (section 6.2).

The Network Identifier is used by CoNet to identify named-data or a service access point. The named-data could be a VDI or a resource referred to in a VDI. A service access point is a network endpoint through which an upper-layer (CoMid or CoApp) entity sends/receives information.

From a user point of view, Network Identifiers are handled as URIs; "inside" CoNet they are transferred inside the CoNet data unit, as specified in D5.1 (section 5.4).

In what follows, we describe how VDI identifiers are mapped to Network Identifiers in the default namespace. We use the following mapping:

VDI identifier: urn:CONVERGENCE:eu:identify-content-server-id:label
Network Identifier: identify-content-server-id/label

where the identify-content-server-id is a unique identifier associated with the CoMid identify content server that released the VDI identifier and label is a unique identifier that differentiates the VDI identified by the server. For instance,

VDI identifier: urn:eu:CONVERGENCE:ics-CONVERGENCE:54ba64ed-a4e0-43e1-a22a-5e37a943ea19
Network Identifier: ics-CONVERGENCE/54ba64ed-a4e0-43e1-a22a-5e37a943ea19

To identify the resource the VDI refers to (e.g. a file), we use a traditional domain-name/path structure directly mapped in a PLHB (Principal Label Hash Based) Network Identifier. For instance:

Plain domain-name/path: test.wim.tv/CONVERGENCE-cedeo/ang.mp4

Network Identifier: test.wim.tv/CONVERGENCE-cedeo/ang.mp4

where “test.wim.tv” is the identifier of the principal of the NID (e.g. the entity that can create NIDs using the *Principal Identifier*, see also section 7.1.3) and the label is “CONVERGENCE-cedeo/ang.mp4”.

4.11 Publish / subscribe pattern

4.11.1 Description

Publish and Subscribe operations are carried out at the middleware level of the CONVERGENCE system. From the middleware perspective, publication of a Resource VDI involves the following steps:

- Creation of a Publication VDI containing
    - Link to the stored Resource VDI (mandatory)
    - VDI Metadata, usually taken from the Resource VDI (mandatory)
    - License regulating access to the publication (optional)
    - Event Report Request defining reports to be issued when specific events occur (optional)
- Injection of the Publication VDI into the overlay of peers forming the “discovery overlay”
- Storage of the Publication VDI on the network

Symmetrically, the subscription process involves:

- Creation of a "Subscription VDI" containing
    - One or more representations of the semantic subscription to a set of metadata, in the form of a SPARQ query or a list of requested metadata (mandatory)
    - License regulating access to the subscription (optional)
    - Event Report Request defining reports to be issued when specific events occur (mandatory)
- Injection of the Subscription VDI into the overlay of peers forming the “discovery overlay”
- Storage of the Subscription VDI on the network

This procedure enables search for subscriptions using standard Search Content procedures (see D5.1 for the Search Content protocol API, see the following chapter for details of Semantic Search Mechanisms), and easy matching of publications to subscriptions. Users are thus given the possibility of revoking their subscriptions and publications, and of extending/altering them at a later time, as with any other VDI. By using VDIs to carry subscriptions and publications, we allow CONVERGENCE to exploit the Event Reporting mechanisms defined in MPEG-21 part 15, embedding Event Report Requests into each

Subscription and Publication VDI. Subscription and Publication VDIs are uniquely identified, and carry the requested metadata plus a reference to the address of the "home" peer of the user to be notified if a match is found. The reference can take the form of a peer identifier or User VDI.

The use of specific licenses for Subscription VDIs and sequences of Subscription VDIs enable many possible extensions. For instance licenses could restrict the kind of information that can be subscribed to by a particular class of users or support focused "forgetting of subscriptions", so that some subscriptions immediately disappear from the system if not satisfied, while other persist indefinitely.

The subscribe operation is carried out by invoking a Subscribe Content Service. Similarly, the publish operation is carried out by a Publish Content Service. Both are complex operations, which involve a chain of Elementary Services. They are thus Aggregated Services. This implies there is no simple Publish Content ES, or Subscribe Content ES, but rather a pub/sub workflow involving coordination of different middleware engines.

CONVERGENCE publication and subscription operations fully comply with the "content based" publish/subscribe paradigm as defined in [19] and support structured semantic descriptions of content. This is much better than "topic based" approaches that require subscribers to know the full name of the content they require.

Semantic subscription makes it possible to store requests for events that change the state of a certain specific VDI (e.g. creation of a new version of the VDI, revocation of the VDI, the creation of a link to the VDI). They also allow subscribers to subscribe to VDIs that have not yet been published e.g. VDIs for:

- New models produced by John
- Special sales of mobile phones at a local store
- Users want to know about all the associations made on columns of a certain author
- Users want to be informed about releases of movies of their favourite actor
- Users want to know how many times other users have done certain things on their content.

In CONVERGENCE, the cases above are equivalent and are handled using the same generic procedures.

When users create a subscription, they issue a (semantic) query. The system returns a list of results. The user can then select the most relevant VDIs from the matching list, and/or wait for results that will be notified later. This requires a system wide request that is universally understandable. It also shows one of CONVERGENCE'S "Unique Selling Points".

Today, when a user subscribes to a particular kind of CD in music.com (e.g. Iron Maiden CDs), she is informed of the CDs only when they are advertised in music.com. In CONVERGENCE her subscription is

system wide and she receives a notification every time anyone publishes information about the CDs that interest her.

Conceptually, the subscription process is split into three parts:

- Part 1: inserting a semantic subscription system wide
- Part 2: matching a subscription once relevant content is published
- Part 3: delivering a notification to the subscriber

4.11.2 Part 1: inserting/storing a semantic subscription system wide

We have two users: P (who publishes) and S (who subscribes). Consider the case in which the subscription is prior to the publication of material matching the subscription.

When S subscribes, she provides semantic metadata, possibly organized in a complex query to filter out unwanted content and group and sort the results. This metadata is semantically validated by the CDS, which helps the user to formulate the query correctly (see relevant paragraphs about the CDS of this same deliverable, section CDS).

S may optionally provide a license governing the use of the information contained in the Subscription VDI.

Along with this metadata, an Event Report Request is created. The ERR defines the event type that will trigger the notification (we call it a “Match” event type) and additional information needed to locate the peer that will receive the notification.

This ERR is inserted into the Subscription VDI. As mentioned earlier, it includes so called "DeliveryParams" (see[26]), that is the address of the peer to be notified if the event occurs. According to the Event Reporting standard, this can be an URI, a CoNet network resource (or service) name, a reference to a User VDI, an email address, etc.

Therefore a Subscription VDI will contain:

1. A structured description of the metadata of the resource the subscriber is interested in
2. A License (defining who can do what with the subscription)
3. An Event Report Request containing
    a. The verb “Match” applied to this Subscription Metadata and any corresponding Publish Metadata

b. The Identifier of the Device or User receiving the Event Report

This set of information is published to the system by injecting its content into the overlay and storing the full Subscription VDI on the network – the same mechanism used for publication.

4.11.3 Part 2: matching a subscription

Later on, when P publishes content, she also provides the (semantic) metadata that best describes the published resource. The metadata is then injected into the system.

When the gossiped information arrives at the destination peers, the procedure is reversed and all pending semantic search operations are performed on these peers, using those data as a target. If a subscription is found (say the subscription by S) that matches the target metadata, it becomes a match candidate. So, whenever a Publication VDI reaches a peer, it is evaluated against outstanding semantic queries from Subscription VDIs known to the peer.

With this approach, we implement a sort of rendezvous node between Subscription and Publication VDIs in the peers responsible for the common Resource Semantic Type (see section 4.7).

4.11.4 Part 3: delivering matches to subscribers

Each match between a new publication and existing subscriptions generates an Event Report - the companion of the ERR embedded inside the matched Subscription VDI. The ER is delivered to the peer/user referenced in the matched Subscription VDI, using an appropriate transport protocol (e.g. the CoNet SendTo primitive is invoked, or an email is sent). When the user (or an application working on her behalf) receives the notification, she can fetch the matching VDIs directly using the Deliver Content protocol.

This solution makes use of the MPEG-21 Event Reporting standard, in conjunction with the concept of a Subscription VDI. In this way, we separate the MPEG-21 event notification mechanism from the internals of injecting metadata to peers. If required, the regular MPEG-21 event notification mechanism can still operate at the client-server level, within the scope of a custom application. To apply the MPEG-21 event notification mechanism system-wide, we use specific Subscription VDIs to inject events. This requires the creation of a new verb (“Match”) in the Rights Data Dictionary (see[26]). Details of this extension to the MPEG-21 standard are left for future deliverables.

4.12 Digital Forgetting

4.12.1 Introduction

In today’s digital world, it is extremely easy for users to publish information via websites, blogs, video-sharing sites, social sites etc, whereupon it comes instantly available to a potentially global audience while search engines and web archiving tools ensure that copies of the information are rapidly

disseminated beyond the site(s) where it was originally published. Private users may also make copies of and republish the information, usually without the knowledge of the original author. As a result, it is practically impossible to locate all copies of the information that has been published, and even harder to remove it, *even when the author is able to remove the information from the site where it was originally published*. There are some circumstances in which this is what the author wishes. For instance, a political dissident may wish to ensure that it is impossible to eradicate her views from the network even if she herself should be forced to demand their removal. However, there are many other circumstances in which a user may legitimately wish to eliminate information she has published. The most commonly cited reasons are personal e.g. the wish to eliminate references to opinions the user no longer holds; the wish to eliminate personal information that has become embarrassing for the user. In other cases, there may be business reasons to withdraw previously published information (e.g. a company's desire to update product information that has been shown to be inaccurate or information on promotional offers that are no longer available). In sum, there is a mismatch between the Internet's tendency to preserve information forever and the needs of users.

4.12.2 Functional Requirements

Against the background described in the previous paragraph, one of CONVERGENCE's goals is to support "Digital Forgetting" of content published on the CONVERGENCE network, allowing users to remove (or update, i.e. remove and re-insert a new VDI), information they have previously published. This goal is translated into formal requirements, which are presented in section 7 of D2.1 and will not, therefore, be readdressed here.

To satisfy these requirements effectively, it is also necessary to meet a number of secondary requirements.

1	The creator of a VDI shall be able to define who can "unpublish" the VDI. The possible options shall include "no-one".
2	Any authorized user shall be able to unpublish the VDI
3	Derived VDIs shall, by default, inherit expiry dates and rights to unpublish from the original VDI, if the publisher does not provide new ones.
4	CONVERGENCE users shall not be able to retrieve an expired or unpublished VDI
5	VDI search and subscribe services shall not return references to expired or unpublished VDIs
6	Expired or unpublished VDIs shall be deleted from the CONVERGENCE network
7	VDIs stored off the CONVERGENCE network shall expose expiry date data to third party applications, allowing users to verify if they have expired (this supports the introduction of

	provisions to make use of expired VDIs illegal)
8	Users shall be able to specify that a VDI has no expiry date (i.e. that it is intended to be "eternal")

4.12.3 Digital Forgetting Mechanisms

To satisfy the requirements defined above, CONVERGENCE will support two forms of Digital Forgetting.

- Pre-planned Digital Forgetting – the CONVERGENCE system defines a default expiry date for all the Publication VDIs published to the CONVERGENCE system. The publisher of a VDI has the ability to override the default value at publication time. He/she can also define who (if anyone) is authorized to change the expiry date. This information is, by default, inherited by derived VDIs. Publication VDIs that have passed their expiry date are no longer retrievable by CONVERGENCE users and are no longer referenced in results from CONVERGENCE search and subscribe services. The associated Content VDI will also be eliminated from CONVERGENCE CoNet.

- On-demand Digital Forgetting – when publishing a Publication VDI, the publisher can define who (if anyone) has the right to unpublish it, (default value: the publisher). The CONVERGENCE system will enable authorized users to request the removal of a Publication VDI, (and associated Content VDI) from the system at any time after publication. The system will then proceed to asynchronously eliminate all stored versions of the Publication VDIs (and their associated Content VDIs) and eliminate all semantic metadata about the VDI, wherever it is stored. Once this operation has been completed, the Publication VDI (and its associated Content VDI) will no longer be retrievable and will no longer appear in search or subscription results. The removal of the Publication VDI will not prevent access to, retrieval and consumption of other Publication VDIs in the same VDI sequence. Other Publication VDIs that declare relationships with the unpublished VDI will remain accessible. However, these relationships will then point to an irretrievable object. On reception of the revocation command, from CoMid, the CoNet performs the appropriate removal actions.

4.12.3.1 Pre-planned Digital Forgetting

During the publishing process, the publisher specifies the time and date at which the VDI will expire. The operational workflows will proceed as follows:

- At Pub VDI publication time:

- The publisher specifies the date and time at which the Pub VDI will expire, which should be before the expiry date of the targeted Resource VDI (Res VDI). If no expiration date is given, the Res VDI never expires
- At the CoMid level:
  - CoMid Publish Content Aggregated Service (PCAS) uses the Overlay TE to gossip the new Publication VDI through the system
  - PCAS submits the Publication VDI to CoNet, for storage and distribution
- At the CoNet level:
  - CoNet stores and disseminates the submitted digital objects as sets of Named-data CoNet Information Units (CIUs, see 7.1) (with a field specifying their expiry date)

- At Publication VDI expiry time (more precisely, from the expiry time onwards):
  - At the CoMid level:
    - The Match TE that handles the matching between Pub and Sub VDIs detects that the corresponding entry in the table is no longer valid and removes it
  - At the CoNet level:
    - Serving nodes and border or internal nodes delete the named-data CIUs in question (those that contain the Pub and Res VDIs), and no longer cache or distribute them

From the Publication VDI expiry time onwards, the CoMid level will no longer contain any trace of the expired Publication VDI and the Publication VDI will no longer be retrievable from the CoNet.

From the Resource VDI expiry time onwards, it will also no longer be retrievable from CoNet.

4.12.3.2 On-demand Digital Forgetting

In this mode of Digital Forgetting, an authorized user specifies his/her desire for the removal (unpublishing) of a Publish VDI for which he/she has “unpublish rights”.

Given that CONVERGENCE incorporates a dynamic set of peers that can join or leave the system at any time, complete and synchronous removal of a Publication VDI from CoMid is not possible. CONVERGENCE therefore adopts an asynchronous approach.

- An authorized user issues a request for the removal (unpublishing) of a Publication VDI.

- At the CoMid level:
  - CoMid Unpublish Content Aggregated Service uses the Revoke Content PE (RCPE), requests relevant peers to eliminate the metadata contained in the Publication VDI
    - The RCPE creates an UnPublish VDI that references a specific Pub VDI. The UnPub VDI declares that its targeted Pub VDI is no longer valid and should be eliminated from match tables.
    - The UnPublish VDI is gossiped throughout the peer collective (employing the Overlay TE) and ends up in the same peers as the corresponding Publish VDI.
    - On receiving the UnPub VDI, peers perform the appropriate "forgetting" activities.
  - The Unpublish Content Aggregated Service issues a revocation command to CoNet, for the deletion of the Pub VDI in question.
- At the CoNet level:
  - On reception of the revocation command from CoMid, the CoNet performs the appropriate removal actions.

Forgetting of the UnPub VDI is handled at the system level. The life cycle of these VDIs is defined as follows.

- At UnPub VDI production/injection time:
  - The UnPub VDI is mandatorily given the same expiry date as that of its associated Pub VDI.
  - At the CoMid level:
    - The CoMid Revoke Content Aggregated Service (PCAS) uses the Overlay TE to gossip the new UnPub VDI throughout the system.

      Given that the UnPub VDI has the same expiry date as that of its associated Publication VDI, it will circulate for as long as there is some possibility that metadata in the Pub VDI is still "circulating" in the system
- At UnPub VDI expiry time (more precisely, from the expiry time onward):

- At the CoMid level:
  - The Match TE detects that the corresponding entry in the table is no longer valid and removes it.

4.13 Real world descriptors

CONVERGENCE needs the ability to describe Real World Objects. However, creating descriptors for all type of objects would be a very major project. Many organizations have produced ontologies or XML schemas for specific domains. For instance, the Association for Retail Technology Standards (ARTS) of the National Retail Federation [42] has developed an XML based schema applicable across multiple retail vertical segments including general merchandise, grocery, convenience, food and drugs. This schema is useful and the CONVERGENCE definition of VDIs for the retailing scenario will "include" and take advantage of these efforts. However the main focus of the retailing scenario is on the exchange of item-related data between systems within the confines of a retail enterprise, and the ARTS scheme does not offer all the features necessary to describe the consumer electronics products, which we will seek to describe in the CONVERGENCE trials. Therefore in this, and in other cases, CONVERGENCE will develop additional domain ontologies. For more information on currently available solutions, see ANNEX A – Survey of solutions of real world descriptors.

4.14 Semantic Search and Content Matching

This section provides a detailed explanation of the *implementation* of semantic search in CONVERGENCE. As in other P2P search overlay structures, the protocol is fundamentally asynchronous. However CONVERGENCE'S publish/subscribe paradigm provides additional decoupling in time and space. Search requests are carried in VDIs. An expiry date dictates whether the user wishes for an immediate reply (as in present-day search engines) or is willing to accept results that arrive in the following hours, days, weeks or months. Aggregation of similar results is carried out by the event report service that notifies the user of relevant matches. Below, we provide a detailed walkthrough of the process.

A CONVERGENCE-compliant Application runs on a random PeerX (maybe a public device in the city hall, or the user's iPad). The application uses services and technologies offered by the CoMid.

0. User UserA, types in some metadata (e.g. resource MOVIE, and genres SCI_FI and CRIME), related to resources she wants to subscribe to.

1. The application invokes the CDS.
   - The CDS expands the request by finding appropriate concepts in known domain ontologies. For example, if the user accepts the IMDB Mapping Movie Ontology [41] (or if the search Application accepts it on her behalf) the CDS maps the user request to the imdb:Movie and imdb:Genre classes.

- The CDS formulates a correct semantic query that reflects the subscription. For example “SELECT ?x WHERE {?x a imdb:Movie . ?x imdb:genres “SCI-FI” . ?x imdb:genres “CRIME”}”, where the prefix imdb: qualifies the IMDB ontology.

- The CDS helps the user to find one (or more) Resource Semantic Types within the CONVERGENCE Core Ontology, and possibly within other semantic dimensions, appropriate for the subscription. For example the user might select the cco:VIDEO semantic type, where cco: is the prefix of the CONVERGENCE Core ontology. In this way, the user exploits the concept of fractals and focuses the search on peers that are interested in the cco:VIDEO type of content, (peers that have previously published or subscribed to cco:VIDEO content and have actively joined the fractal). The CDS prepares an incomplete metadata tag describing the Resource Semantic Type of the new Subscription VDI (the VDI identifier is still not known).

2. The application invokes Create Content.

   - A Subscription VDI is created which contains:

     - A unique identifier (SVDI_1)

     - The completed metadata indicating the core semantic type of the VDI, i.e. an RDF triple such as {SVDI_1 cco:hasReST cco:VIDEO}, embedded inside a didl:Descriptor tag.

     - The requested query, inserted into another didl:Descriptor tag

     - License and ERR

     - Expiry Date of the VDI.

   - The ERR says "notify peer PeerA if match". PeerA is the "home" peer of UserA. It is a peer that will receive the match notifications (the Event Reports).PeerA runs the Store Event and the Request Event servers "of the user".

3. The application invokes Inject Content (see Figure 25).

   - If PeerX is not yet a member of the cco:VIDEO fractal, it registers to it and joins.

   - Information about the Subscription VDI is circulated in the fractal. Thus, the Subscription VDI SVDI_1 reaches the peers that participate in the overlay fractal assigned to it, i.e. the fractal named “VIDEO” (a fully qualified name would be in the form `urn:overlay:registry:cco:video`).

   - Each peer extracts information contained within the Subscription VDI (the identifier of the VDI, the embedded query and any additional metadata, license, ERR) and copies it to its own “Subscriptions Table”. The “Subscriptions Table” keeps track of all Subscription VDIs that reach the peer, and indexes them using their VDI identifier.

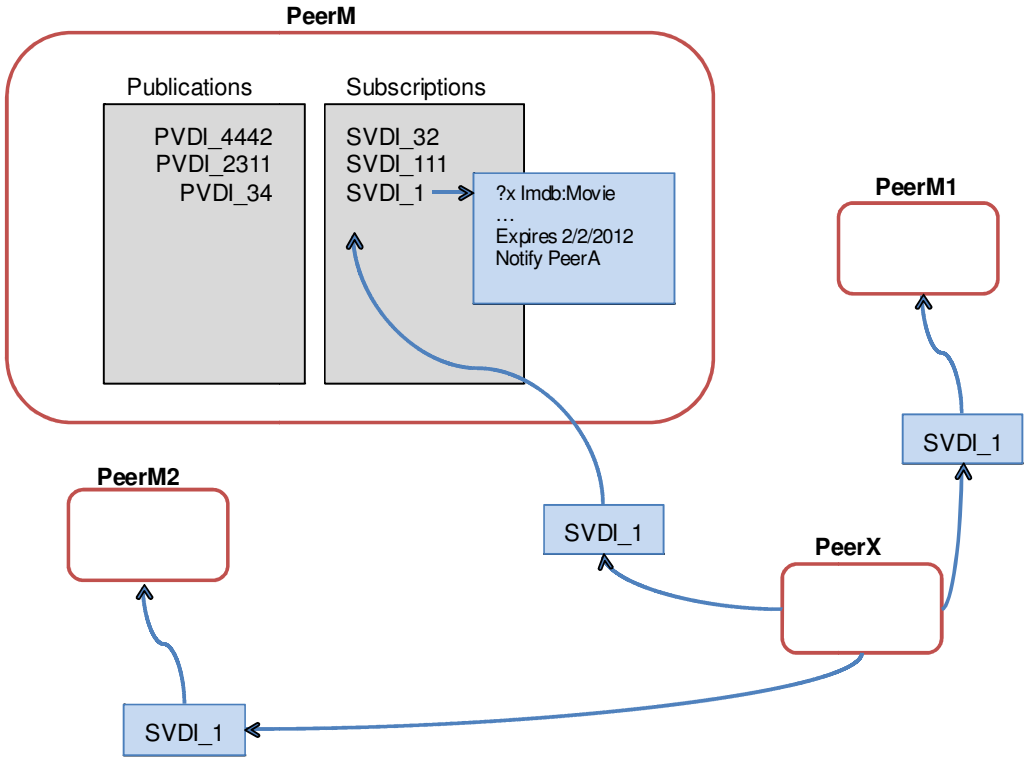


Figure 25 – Semantic Subscription SVDI_1 reaches PeerM, PeerM1 and PeerM2 of the VIDEO fractal

4. The application invokes Store Content.

    - The Subscription VDI is stored by CoNet as a generic network resource (just like any other VDI). This means that knowledge about the subscription is no longer restricted to the CoMid and can be **made available to crawlers that are not based on CoMid** (subject to security restrictions enforced by CoNet).

Another user makes a publication at a later time. That is: on a random PeerY (see Figure 26):

0. A user decides to publish a resource (e.g. a movie) for which she (or somebody else) has already created a Resource VDI. The Resource VDI describes the movie and the license that regulates access to it. The Publication VDI copies those Descriptors and adds an expiry date and license. Metadata describing the resource is collected from its Resource VDI, and possibly refined by the user. The user now has the role of a publisher. We assume the author of the resource has given her permission to publish it.

    - RDF triples describing the movie are extracted from its Resource VDI RVDI_23. The movie has been described using an alternative domain ontology about movies (e.g. the Movie Ontology at www.movieontology.org).

- The following RDF Descriptor tags, taken from the original Resource VDI, are inserted into the new Publication VDI:
    - {RVDI_23 rdf:type movientology:Movie}
    - {RVDI_23 movientology:belongsToGenre movientology:Sci-Fi}
    - {RVDI_23 movientology:belongsToGenre movientology:Thrilling}
    - {RVDI_23 movientology:belongsToGenre movientology:Actionreach}

  The subject of the above semantic relationships is obviously the original resource VDI. They are now inserted into the new VDI, using CONVERGENCE semantic links. A specific tag allows RDF fragments to describe a VDI. In the case of publications, such VDIs simply state that they carry information about other VDIs.

1. The application invokes the CDS.
    - The CDS helps the user find one (or more) Resource Semantic Types within the CONVERGENCE Core Ontology or other shared semantic dimensions that provide appropriate classifications for the publication. For example the user selects the cco:VIDEO semantic type
2. The application invokes Create Content.
    - A Publication VDI is created that contains:
        - A unique identifier (PVDI_1)
        - The original didl:Descriptor metadata tags, now “describing” the Publication VDI
        - An RDF metadata triple describing the core semantic type of the Publication VDI and thus indicating the Publication VDI’s destination fractal in the overlay (this information is embedded in another didl:Descriptor)
        - An explicit reference to the Resource VDI the Publication VDI refers to, i.e. {PVDI_1 cco:isPublicationOfRVDI_23}
        - Optionally a license and an ERR, if the publisher wants to limit discoverability of the publication or to be notified when the publication is matched
        - Expiry Date of the VDI.
3. The application invokes Inject Content
    - If PeerY is not yet part of the cco:VIDEO fractal, it registers to it and joins.
    - Information about the Publication VDI is circulated: PVDI_1 reaches the peers that participate in the overlay fractal assigned to it, i.e. the “VIDEO” fractal.

- Each peer extracts information contained in the Publication VDI (the identifier of the VDI, all metadata, license, ERR) and copies it to its own "Publications" table. The "Publications" table keeps track of all Publication VDIs that reach the peer, and indexes them using their VDI identifiers.

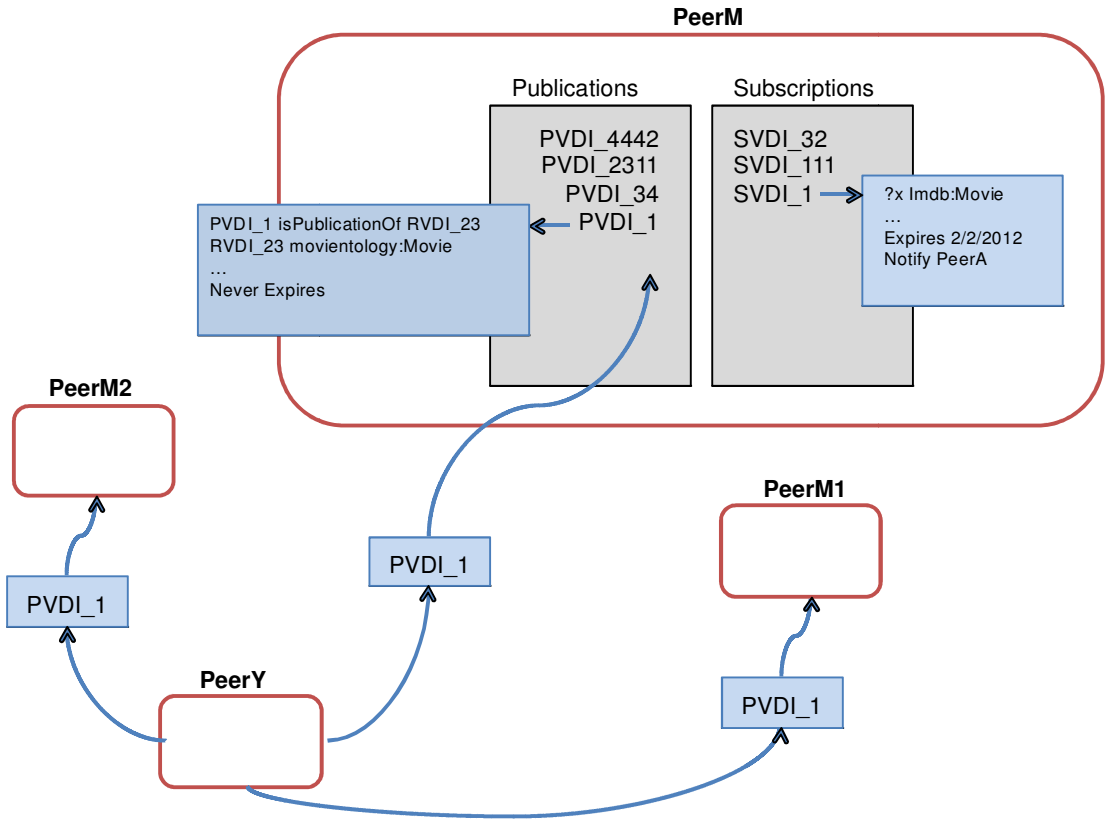


Figure 26 – Publication PVDI_1 reaches PeerM, PeerM1 and PeerM2 of the VIDEO fractal

4. The application invokes Store Content.

   - The Subscription VDI is stored by CoNet as a generic network resource generic network resource (just like any other VDI). This means that knowledge about the subscription is no longer restricted to the CoMid and can be made available to crawlers that are not based on CoMid (subject to security restrictions enforced by CoNet)

The publication and the subscription VDIs reach several rendez-vous peers in the VIDEO fractal (see Figure 27):

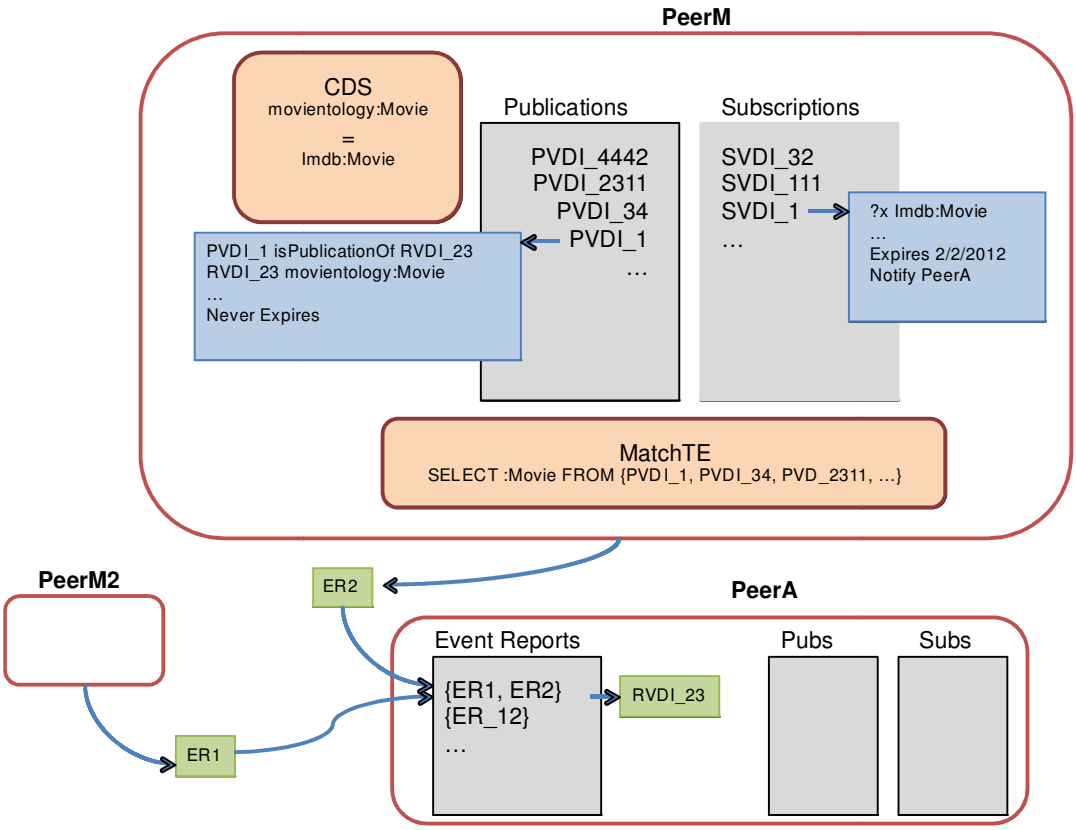


Figure 27 – Subscription is matched by PeerM and PeerM1

5. Match TE constantly monitors peer tables and is triggered on arrival of a Publication VDI or a Subscription VDI. The goal of the Match TE is to perform a match between the SPARQL queries embedded inside the Subscription VDIs stored in the "Subscriptions" table, and the metadata stored in the peer's "Publications table".

   - When a Publication VDI arrives at the peer, the full list of SPARQL queries in the "Subscriptions" table is scanned, and run against the newly arrived metadata.
   - When a Subscription VDI arrives at the peer, the embedded SPARQL query is run against the full list of metadata found in the "Publications table".

   Before attempting to match a SPARQL query against the available data, the system evaluates possible semantic equivalences between concepts belonging to different ontologies.

   - Match TE invokes CDS.

- The CDS asserts a semantic equivalences between the imdb:Movie **class** and the movieontology:Movie **class,** between the movientology:belongsToGenre **property** and the imdb:genres **property,** and between the movientology:Sci-Fi **individual** of type movientology:Genre and the "Sci-Fi" **literal** assigned to the imdb:genres property[5]. The task of finding equivalent genres is greatly facilitated by the preliminary assertion of the equivalence of associated properties (preceding bullet), and of course matching of the literal with the individual name.

If more than one SPARQL query is to be executed, the Match TE runs a query engine capable of processing semantic queries. The target data for the query is the aggregate of all metadata relevant to the query, after query expansion of the semantic classes. Match TE asks what are the RDF triples that satisfy the query and fetches them.

If matching RDF triples are found, a "Match Event" is triggered. The ERR is extracted from the Subscription VDI, the address of PeerA is read and companion ER is generated.

The ER is filled with the identifiers of all Resource VDIs known to PeerM, which contain matching metadata RDF triples.

6. Match TE invokes Event Report TE

   - The ER is sent to PeerA using the transport protocol specified in the original ERR, and invoking PeerA's Store Event service.

Different peers in the VIDEO fractal send Event Reports, notifying successful match events, and deliver them to PeerA. Note that some of the peers of the fractal may not be able to infer semantic equivalences because not all CDS engines have the proper dictionary to translate between the movieontology: and imdb: ontologies.

0. Since PeerA may receive several match notifications from different peers of the fractal, the peer performs a "notification fusion" eliminating duplicate VDI ids matching the subscription.

   When the user wants to check whether her home peer has received new notifications, she polls the Request Event service of PeerA from PeerX, where she is currently located. Alternatively, if PeerA is the user's mobile/laptop of the user, a GUI alert pops when the ER is received.

4.15 CoMid Security

This section details the methodology introduced in section 3.4, starting with the identification and authentication of CONVERGENCE entities.

[5]The IMDB ontology contains a detailed taxonomy of genres in the form of a class hierarchy. The Movieontology allows representation of genres through string literals.

4.15.1 Roles

CoMid security provides technical means for handling the identification and authentication of CONVERGENCE entities. In this section, we define what we mean by identification and authentication, we identify the actors (roles) responsible, define the assets to which they apply, and analyze threats to these assets. We go on to define objectives that need to be reached to defend against these threats, and, on this basis, specify functional requirements for CoMid security.

We begin by defining identification and authentication:

Table 1 – Definition of Identification and Authentication in CONVERGENCE

Term	**Definition**
Identification	This is the process of assigning an identifier to an entity (e.g. a user, end-user, device, etc.). Identification is usually considered to be a sub-process of registration, and is performed once for each entity prior to the entity performing any action.
Authentication	This is the process of proving a claimant's identity. Authentication is a regular procedure usually required prior to processing requests from a claimant.

Note: The first definition is taken from the MPEG environment and deviates from common definitions of "identification", which often comprise authentication (proof of identity) or determination of an identity.

In CONVERGENCE scenarios, the categories of actor responsible for handling identification and authentication are **Identity Providers** and **Service Providers**. Each occupies a specific **role** in CoMid security.

Identity Provider

The Identity Provider is responsible for the registration of users. This may involve:

1. Identification, i.e. assignment of identifiers; including issuance of certificates
2. Verification of credentials
3. Enforcement of policies
4. Revocation/Renewal of identities

Depending on the scenario, registration can be a simple assignment of an identifier from a name space, or an extensive procedure involving validation of personal data and additional information.

The Identity Provider is a trusted third party, usually independent of users. For example, in a university scenario, the Identity Provider might be a commission made up of professors; in a governmental application it might be a government agency.

The Identity Provider does not normally interfere with the "regular" daily business of using the identifiers assigned to entities.

Service Provider

The Service Provider is in charge of "daily" business involving users, entities, etc. for a specific scenario. Apart from its general obligations, depending largely on the underlying application, the responsibilities of a Service Provider may include:

1. Authentication, i.e. verification of claimant's identities, and proof of its own identity
2. License issuance
3. License validation

Although the two roles – Identity Provider and Service Provider – may coincide for special applications, they are normally strictly separated. Thus, an Identity Provider may seek private information from users that it does not reveal to Service Providers. Conversely, Service Providers do not reveal the detailed data they collect to Identity Providers. For instance, they will not reveal detailed profiles of customers' shopping behaviour, especially when they are competing with other Service Providers on the same market.

4.15.2 Assets

Assets may include:

a. Assets related to VDIs
b. Assets related to users (end-users, peers, etc.) and services (service applications)
c. Assets related to engines (software packages) in the CONVERGENCE middleware.

The CONVERGENCE framework identifies the assets listed below.

Table 2 – CoMid security Assets

No	ID	Asset	Description
1	ASS.VDI-ID	VDI Identification	Each VDI is assigned a unique identifier. A VDI identifier is never used twice, not even for two related or "updated" VDIs.
2	ASS.VDI-INT	VDI Integrity	A VDI comprises components such as content, metadata, annotations, *inter alia*. The VDI's unique identifier is also considered one of its components. The integrity of a VDI therefore refers to the VDI as a whole, including its identifier. A VDI should never be changed, once it has been published, until possible revocation. The identifier must be unique for each VDI, in particular the identifier "Anonymous" is not allowed.
3	ASS.VDI-AUTH	VDI Authenticity	Beyond integrity, VDI authenticity refers to the origin of a VDI. VDI authenticity describes the assurance for a user that the VDI has indeed been created by the entity claiming its creation.
4	ASS.USER-ID	User Identification	A user may be a human end-user, or a device, or a network peer. Like a VDI, a user is identified by assigning an identifier to her. Identifiers are unique in the sense that no two users are allowed to share the same identifier. Unlike the situation for VDIs, one and the same user may – under certain circumstances – hold different identifiers at the same time. Even the identifier "Anonymous" can be supported, depending on the underlying policy.
5	ASS.USER-AUTH	User Authenticity	Users authenticity refers to assurance of the "integrity" of a user at the moment of his or its identity claim. In other words, the authenticity of a user claiming a specific identity describes the fact that the claimant is indeed the one to

			whom the identity has originally been assigned.
6	ASS.SERV-AUTH	Service/Application Authenticity	Service (Application) authenticity defines the integrity of the service at the moment it claims a given identify. Any user (in particular human end-users) can – at his or her discretion – gain assurance that a service claiming a specific identity is indeed the one to which this identity has been assigned.
7	ASS.ENG-ID	Engine Identification	This asset refers to the identification of an engine.
8	ASS.ENG-AUTH	Engine Authentication	This asset refers to the authenticity of an engine, here meaning its integrity and possibly origin.
9	ASS.ENG-VDI	VDI Technology Engines	This asset refers to Technology Engines handling VDIs.
10	ASS.TE-LIC	License Technology Engine	This asset refers to licensing technology, referring both to issuance and compliance.
11	ASS.TE-ER	Event Reporting Technology Engine	This asset refers to monitoring and reporting of events.
12	ASS.SEC	Security Technology Engine	This asset refers to software and hardware handling cryptographic services, and interfacing CoSec.

4.15.3 Threats

The CONVERGENCE framework will be designed to address the following security threats.

Table 3 – CoMid Security Threats

No	ID	Description
1	T.VDI-ID-NONE	A VDI may not have been assigned an identifier prior to publishing.
2	T.VDI-ID-MULT	One and the same VDI may be assigned multiple identifiers.

3	T.VDI-ID-NONUNIQUE	A VDI may be assigned an identifier that has already been assigned to another VDI.
4	T.VDI-ALTERED	Components of a VDI may be altered after publishing.
5	T.VDI-ID-REPLACE	The identifier of a VDI may be exchanged with another identifier, the remaining content remaining unchanged.
6	T.VDI-CONTENT-REPLACE	The (entire) content of a VDI may be replaced or altered while keeping its identifier. **Note**: T.VID-ID-REPLACE and T.VDI-CONTENT-REPLACE are special cases of T.VDI-ALTERED. The identifier of a VDI is considered to be a component of the VDI.
7	T.VDI-AUTH-REMOVE	The origin of a VDI may be removed, making it impossible to determine its creator.
8	T.VDI-AUTH-ALTER	The origin of a VDI may be altered, suggesting that the VDI has been created by an entity different from the one that really created it.
9	T.USER-ID-NONE	A user does not receive an identifier although – by the policies in place – he or she is required to hold one.
10	T.USER-ID_MULT	A user may be assigned multiple identifiers in a situation where unambiguous identification is a requirement.
11	T.USER-ID-AMBIG	A user may be assigned an identity that has already been assigned to another user.
12	T.USER-AUTH-FORGE	A user tries to authenticate himself or itself under a “false” identity that has not been assigned to him (i.e. it has been assigned to someone else, or not assigned to anyone at all).
13	T.SERV-AUTH-FORGE	A service (application) tries to authenticate itself under a “false” identity, namely an identity which has not been assigned to it.
14	T.SERV-AUTH-MISS	A service fails to authenticate itself, although it has explicitly been requested to do so by another user, or is implicitly required to do so under policies in force.

4.15.4 Objectives

The CONVERGENCE framework will seek to achieve the following security objectives

Table 4 – CoMid Security Objectives

No	ID	Description
1	O.VDI-ID-UNIQUE	If each VDI is to obtain a unique identifier, assigning identifiers cannot be the responsibility of VDI creators. Instead, servers will draw these identifiers from disjoint name spaces and assign an identifier to each VDI before publishing. Following publishing, no VDI may be altered in any way prior to revocation.
2	O.VDI-INT	The integrity of a VDI is regarded as critically important. A VDI must remain unaltered throughout its lifetime, i.e. after publishing and before revocation. The requirement that the VDI should not be altered applies to the whole including, in particular, its identifier, and components such as metadata, references, annotations, and content.
3	O.VDI-AUTH	Where required or granted by the creator, the origin of a VDI must remain unchanged throughout its lifetime. In such cases, a user shall be able to gain assurance about the authenticity of a VDI. **Note**: depending on the policies in forces – users may be allowed to create VDIs anonymously, or on behalf of a group of users, or under a pseudonym.
4	O.VDI-AUTH-ALTER	It shall be impossible to alter the origin of a VDI, as stated at the time when the VDI was created. If it is not possible to achieve this objective, it should be possible to *detect* any alteration to the origin of the VDI. This means that if an origin is claimed, it shall be declared in a way that can be verified by an independent third party.
5	O.VDI-AUTH-REMOVE	It shall be impossible to remove the origin of a VDI, as stated at the time when the VDI was created. If it is not possible to achieve this objective, it should be possible to *detect* the removal of the origin.
6	O.USER-ID	There shall be a service responsible for assigning a unique identifier to each user. This process shall be part of a registration procedure, governed by an independent third party acting as an Identity Provider. **Note**: The interpretation of the registration procedure may vary considerably for different scenarios, ranging from simple identifier assignment from a name space to governmental procedures

		encompassing passport or even biometric validation.
7	O.USER-AUTH	Each user shall be able to authenticate her claimed identity to an independent third party. Note however a user can hold any one of four types (categories) of identity (F.USER-ID-SINGLE, F.USER-ID-GROUP, F.USER-ID-PSEUDONYM and F.USER-ID-ANONYMOUS). The identities used in any specific scenario will depend on the characteristics of the scenario.
8	O.SERV-ID	Unlike users, services (applications) shall always have a unique (non-anonymous) identification.
9	O.SERV-AUTH	A server shall be able to authenticate to a user at any time. A server shall authenticate to a user at the user's discretion, or if required by underlying policies.

4.15.5 Security Requirements

On the basis of the threats described above, we have identified the security requirements listed below.

Table 5 – CoMid Security Requirements

No	ID	Description
1	F.USER-ID-UNAMBIG	The Identity Provider shall ensure that no identifier is used twice for different users. **Note:** This requirement does not conflict with the goal of supporting group identities. In this case, the group is considered as the user. In some cases, depending on policies, it may be admissible to assign multiple identifies to the same user.
2	F.USER-ID-UNIQUE	If required by policy, the Identity Provider shall be able to ensure that each user receives a unique identifier.
3	F.USER-ID-MULT	If allowed or required by policy, a user may have multiple identities: a. With different Service Providers, if they so allow b. With the same Service Provider, if it so allows
4	F.USER-ID	To authenticate to and access specific services of a CONVERGENCE instance, a user must first be identified by an Identity Provider. Identity providers can authenticate the following categories of user identity: and F.USER-ID-ANONYMOUS.

		a. An individual/organization identity (F.USER-ID-SINGLE) b. An identity certifying membership of a group (the identity of individual can only be revealed by the group administrator) (F.USER-ID-GROUP) c. A context dependent Pseudo-identity (F.USER-ID-PSEUDONYM) d. An identity providing complete anonymity (F.USER-ID-ANONYMOUS)
5	F.USER-ID-SINGLE	A user (end-user or organization) shall be able to authenticate himself/herself/itself with a single, unique, individual identity confirmed by an Identity Provider. This type represents the "ordinary" way an identity is used. The normal way in which user confirms this kind of identify is to provide proof of possession of a specific secret. Where policies allow, an individual user may have multiple single identities
6	F.USER-ID-GROUP	A user shall be able to register as a member of a specific group of users. This registration is carried out by the Identity Provider that sets up the group. In this case, the user shall be able to authenticate himself/herself/itself as a legitimate member of the registered group, while remaining entirely anonymous within the group. *This type of identity is supported by cryptographic procedures known as group signature schemes. These schemes allow members of a group to issue signatures on behalf of the entire group, while remaining anonymous. The same schemes allow the group administrator to unveil and/or revoke the identity of an individual group member, for instance, when this is required by legislation.*
7	F.USER-ID-PSEUDONYM	Each user shall be able to hold context-specific pseudonyms, and authenticate himself/herself using the pseudonym appropriate for a given content. A user's pseudonyms shall be unique within each context, but different and not linkable across different contexts. Pseudonyms are generated according to a key scheme set up, and are not at the discretion of the users. This type of pseudonym is supported by the "Restricted Identification" technique deployed in the German Electronic Passport.
8	F.USER-ID-ANONYMOUS	A user shall be able to act under an entirely anonymous identity, making his/her appearances absolutely indistinguishable from those of any other user, including his/her own previous

		appearances. In other words, the identity “Anonymous” shall be considered as “void”.
9	F.USER-AUTH-CONV	To access the specific services provided by a CONVERGENCE instance, a user shall be authenticated, using one of the four types of identities listed under F.USER-ID.
10	F.USER-AUTH-SERV	Users may need to be further authenticated by a service application, depending on the specific requirements of the application.
11	F.SERV-ID	Each application service shall be identified by one or more Identity Providers with a trust relationship with the entity running the CONVERGENCE instance. Each application shall receive a unique (non-anonymous) identifier.
12	F.SERV-AUTH	A user shall be able to request authentication of an application. An application shall be able to authenticate itself in response to such a request, and may be required to do, depending on policies.
13	F.PEER-AUTH	It shall be possible for peers to mutually authenticate each other.
14	F.ENG-ID	Identification is a CONVERGENCE-governance issue and ensures that any identifier is unique. Engines shall be identified by an Identity Provider.
15	F.ENG-INT	Engines shall have the ability to verify (i.e. check the integrity of) an engine.
16	F.ENG-AUTH	When a peer downloads an engine, it shall authenticate the engine before installing it. **Note**: Assuring correct download and installation of engines is not enough to guarantee they will operate properly for the lifetime of the device where they are installed. However, device integrity is far too complex a matter to be handled by CONVERGENCE. We therefore assume that engine software will be operated on a trustworthy device. Using a smart card as a trusted hardware security module can partially mitigate this problem.
17	F.ENG-VDI	The VDI TE shall have the ability to: a. *Identify* a VDI and its components b. *Sign* a VDI and its components (*individual* and *group signature*) c. Make parts of a VDI accessible only to certain users

		(*access restriction*) d. *Authenticate* the creator of a VDI e. *Verify* a VDI and its components (verify the integrity) **Note**: c. will usually require encryption. If many or most VDIs are encrypted, the main challenge will not be encryption itself but key escrow and, more generally, key management.
18	F.ENG-VDI-SIGN-THPART	A user may use the services of a trusted third part to sign a VDI. **Note**: Signatures can allow tracking and profiling of users, even when VDIs are encrypted. This is a privacy concern. The requirement to allow signatures by trusted third parties is designed to address this concern.
19	F.TE-LIC-ISS	The License TE shall have the ability to create a license whose principal is a user as defined above.
20	F.TE-LIC-ENF	The License TE shall have the ability to enforce a license, i.e. force a peer to comply with rights included in a license.
21	F.TE-ER	The Event Reporting TE shall have the ability to force a peer to issue a message to the users and peers specified in the Event Report Request (ER-R) whenever the peer executes one of the verbs specified in the ER-R.
22	F.SEC	The Security TE shall have the ability to a. Encrypt/decrypt i. A resource using a *symmetric* key ii. A key using an *asymmetric* key (by CoSec) b. Create new *credentials* and manage *certificates* c. Generate *symmetric* keys Store confidential information e.g. licenses and keys in the secure repository Through the interface to CoSec , it will provide: a. Generation of asymmetric key pairs b. Management of group signature scheme c. Management of Identity Based Encryption Schemes d. Management of Attribute Based Encryption Schemes e. Management of pseudonyms

4.16 CoNet Security

CoNet security and privacy mechanisms are decoupled from the mechanisms adopted at the middleware level (CoMid).

In this section we provide a preliminary description of security and privacy issues of specific concern to CoNet. The discussion is limited to security/privacy assets, threats and objectives.

As far as concerns security, we plan to implement functionalities that achieve the security objectives defined later in this section. As far as concerns privacy, we plan to devise a CoNet architecture that does not prevent the implementation of privacy preserving mechanisms. However, our implementation of the CONVERGENCE framework will not include such mechanisms.

From a high-level point of view, we aim to protect the following CoNet security and privacy assets:

- *Service integrity*, i.e. access to valid named-resources
- *Producer/user anonymity*

in cases where:

- The Infrastructure cannot trust users
- Users cannot trust the infrastructure
- Nodes in the infrastructure cannot trust each other.

We observe that, at least currently, it is not planned that CoNet should support *confidentiality,* first because this is primarily an end-to-end issue, second because. CoMid already provides *confidentiality* support. The final decision on this issue will be described in future deliverables.

The table below describes security and privacy assets, threats and objectives relevant to CoNet.

Table 6 – CoNet security and privacy assets, threats and objectives

(security asset 1.1)	Integrity of name-system routing entries
(security asset 1.2)	Integrity of name-system responses
(security asset 2.1)	Integrity of named-data
(security asset 2.2)	Originality of named-data
(privacy asset 1.1)	Anonymity of the producer of named-data
(privacy asset 1.2)	Anonymity of users who request named-data
(security threat 1.1)	An adversary remotely accesses the routing database of the name-system-node
(security threat 1.2)	An adversary intercepts and forwards packets belonging to a communication between a node (border and end-node) and the name-system-node

(security threat 2.1)	An adversary generates a fake version of named-data
(security threat 2.2)	An adversary intercepts, alters and forwards named-data
(privacy threat 1.1)	The behaviour of a user/producer is monitored and profiled by sniffing CoNet packets. Sniffing could occur in any part of the network, with the exclusion of the local access network (e.g., ADSL link, WLAN, etc.), which is trusted by the user
(security objective 1.1)	CoNet shall protect the routing database of the name-system from unauthorized modifications by remote users, i.e. Insert, modify and delete entries
(security objective 1.2)	CoNet shall avoid impersonification of the name-system and the altering of name-system responses
(security objective 2.1)	CoNet shall avoid the caching of fake versions of named-data
(privacy objective 1.1)	CoNet shall not prevent the implementation of mechanisms that allow verification of the integrity and originality of named-data, without disclosing the identity of the user that has produced the named-data
(privacy objective 1.2)	CoNet shall not prevent the implementation of mechanisms that make it possible to deliver named-data without disclosing the identity or network location of the requesting user to intermediate un-trusted nodes.

4.16.1 CoNet Service Integrity

In an Information-Centric network, several kinds of malicious attacks could jeopardize access to a named-resource. To identify possible security threats, we adopt an architecture-centric approach, considering the different functionalities provided by the CoNet architecture, and for each functionality identifying potential threats. Obviously, this initial model of security threats will be extended as the project proceeds.

4.16.1.1 Routing-by-name

CoNet routing-by-name is carried out using the *lookup-and-cache* approach described in D5.1 and in section 7.1 of this report. Briefly, nodes use the name-based routing table as a *routing-cache*. If a routing entry is missing, the node looks up the routing entry on a name-system.

To preserve the routing-by-name mechanism, the assets we need to protect are:

- (Security asset 1.1) Integrity of name-system routing entries
- (Security asset 1.2) Integrity of name-system responses

We assume that an adversary may:

- (Security threat 1.1) have the ability to access the routing database of the name-system remotely
- (Security threat 1.2) intercept and forward packets of a communication between a node (border and end-node) and the name-system

We further assume that remote access to the name-system is always guaranteed, i.e. that there are mechanisms in place to prevent DoS attacks, for instance, saturation of the name-system with a huge number of requests.

To preserve assets against these threats, CoNet security objectives shall include the ability to:

- (security objective 1.1) protect the routing database of the name-system from unauthorized modifications (insert, change, delete entries) by remote users
- (security objective 1.2) avoid impersonification of the name-system and the altering of name-system responses

4.16.1.2 In-network Caching

A fundamental feature of CoNet is the ability of network nodes to cache chunks of named-data, i.e. named-data CIUs. Any border or internal node can cache chunks of named-data that it is forwarding, and can replay a cached chunk of named-data on request. This possibility offers undeniable performance benefits. However, it creates the risk that fake versions of named-data could prevent the retrieval of the original version. A fake version of named-data is a version consisting of named-data with the same name (NID) as the original, but different content. If an intermediate node (border or internal) has unintentionally cached a fake version of named-data, it will respond to requests for the original named-data with the fake version, preventing end-nodes that make such requests from downloading the original data.

To preserve the possibility of retrieving the original version of named-data, the security assets we need to protect are:

- (Security asset 2.1) Integrity of named-data; i.e. the certainty that named-data has not been modified since they were made available by the author
- (Security asset 2.2) Originality of named-data; i.e. the certainty that named-data has been created by the original author, that is the user with the right to use the name (e.g. www.cnn.com) of the named-data.
-

An adversary may:

- (Security threat 2.1) Generate a fake version of named-data
- (Security threat 2.2) Intercept, alter and forward original named-data

However, for the scope of this work, we assume that cached named-data cannot be manipulated, i.e. the cache storage unit is trusted.

In order to preserve in-network caching assets against these threats, CoNet security objectives will include the ability to:

- (Security objective 2.1) avoid caching of fake versions of named-data.

4.16.2 CoNet Producer/User Privacy

The CoNet Information-Centric networking paradigm poses new challenges for user privacy. For instance, protecting the integrity and originality of cached named-data might lead to disclosure of the identity of the author. Similarly, the source-routing approach used by CoNet to deliver named-data to a requesting user might disclose the network location and thus the identity of the requesting user. This said, the privacy assets we should protect are:

- (Privacy asset 1.1) the anonymity of the author of named-data
- (Privacy asset 1.2) the anonymity of the user who requests named-data

There are several possible threats to these assets:

- (Privacy threat 1.1) an adversary could monitor and profile the behaviour of a user by sniffing CoNet packets.

At the time of writing, we assume that packet payload is encrypted at the CoMid level and is therefore not accessible to an adversary. However, this assumption could be removed in future deliverables. To preserve privacy assets against this threat, the CoNet architecture should not prevent the implementation of privacy preserving mechanisms. The objectives of such mechanisms would be:

- (Privacy objective 1.1) to verify the integrity and originality of named-data without disclosing the identity of the user that produced the data
- (Privacy objective 1.2) to deliver named-data without disclosing the identity or the network location of the requesting user to intermediate un-trusted nodes

4.17 Scalability

Technical testing of the CONVERGENCE architecture will take place in special test-beds with restrictions on numbers of users, volumes of content and transaction rates. The scalability of the CONVERGENCE architecture can be defined as the possibility of:

1) Deploying the system architecture in real life scenarios, under a predictable initial load
2) Further expanding the system as additional users and content are added in the future.

In this section we describe the **conceptual framework** we will use to assess the scalability (in WP8) of the CONVERGENCE architecture, with reference first to the CoMid and then to the CoNet.

4.17.1 CoMid scalability

CONVERGENCE content is described using Metadata included in VDIs. To be scalable, the CoMid has to allow an arbitrarily high number of users (all users of the current and the future Internet) to publish and retrieve an arbitrarily high number of VDIs. Given that publish-subscribe operations in the CoMid are implemented in a distributed way, by a set of CONVERGENCE peers, we can characterize the scalability of CONVERGENCE in terms of relations between the input load and the capacity of the CONVERGENCE system, assuming the need to meet a set of performance targets. The input load can be characterized by: i) number of published VDIs; ii) number of active subscriptions; iii) overall rates of publication, subscription and search operations (e.g. the product of the number of users and the publication/subscription/search rate for each user). In addition, we need to quantify the volume of "indexable" metadata contained in each VDI and the complexity of the queries used in search and subscribe operations. The capacity requirements for the CONVERGENCE system and for CONVERGENCE peers are given by: i) the number of peers; ii) the storage capacity of each peer; iii) the processing capacity of each peer; iv) the inter-peer communications load (i.e. the messages that peers need to exchange to support user requests). Finally, it is possible to set system performance targets in terms of: i) lookup time for search operations; ii) relevance of search results and notifications. In this conceptual setting, CONVERGENCE may be considered to be scalable if its capacity requirements are a linear or low order polynomial function of input load.

In practice, a full quantitative analysis of CONVERGENCE scalability may not be possible. In what follows we will therefore adopt a qualitative approach.

The semantic overlay implements load-critical operations (searching, indexing in pub/sub tables and circulation of metadata etc) in the CoMid, which is designed with scalability in mind. The design takes account of a number of critical conditions.

- **High network dynamism**. Peers can join and leave the network at any time. The search procedure has to perform normally, even if a peer fails.
- **High content dynamism**. Content stored in the network is highly dynamic due to frequent publications and updates of previously published contents.
- **Large search space.** Search and subscribe operations need the ability to find any publication on any peer in the CONVERGENCE network.

Given the need for scalability the design should have the following goals (see also [39]).

- The Semantic overlay must be **decentralized**: there should be no central entity controlling the search procedure. This avoids single-point-of-failure issues and ensures load-balancing between the peers that take part in the procedure.

- Semantic overlay must be **efficient**: peer resources and network bandwidth should be used with moderation.
- The Semantic overlay must be **scalable**: efficiency should be maintained when the volume of published content increases
- The Semantic overlay must be **fault-resilient**: if a peer fails, search should be carried out normally.
- The Semantic overlay must be **load-balancing**: the resources used for search should be distributed across all the peers of the network.

CONVEGENCE has to deal with highly heterogeneous metadata based on different vocabularies and with ambiguous and inaccurate descriptions. To minimize the search space, search is restricted to publications of a single semantic type, and signalling is propagated only to a subset of the overlay (the relevant fractal). However, CONVERGENCE also needs to address an additional requirement:

- Semantic overlay must be **flexible**: search queries provided by the user should be treated in a semantic manner, ensuring that results match the 'meaning' of the query.

These requirements directly affect scalability: query expansion and the efficiency of SPARQL query engines needs to be evaluated in terms of their impact on the overall scalability of the CONVERGENCE system.

Further details on support for scalability can be found in the description of the propagation protocol for the semantic overlay (chapter 4.6.3).

4.17.2 CoNet scalability

As far as concerns CONVERGENCE'S Information-Centric aspects, the CoNet has to scale up to the scale of the current Internet, i.e. it should be capable of replacing the storage and retrieval capabilities of the current Internet.

To assess the scalability of CoNet, we use a conceptual framework similar to the framework we used to analyze the CONVERGENCE publish-subscribe model. This means we need to define: i) the input load, ii) the capacity requirements for the system, and iii) performance targets and characterize the relations between them. In this setting, the input load is defined by i) the number of different addressable items; ii) the number of *Principal Identifiers* (see sections 4.10 and 7.1.3); and iii) the rate of user content requests. System capacity is defined in terms of i) storage and processing requirements for different types of CoNet nodes, ii) inter-node traffic to be supported. Performance targets will include i) content download rates (for non real-time content); and ii) loss and delay. Some details of our analysis will be

included in the sections of this report dedicated to the design of the CoNet (e.g. Name-based routing: lookup-and-cache). Other details will be reported in future deliverables (of WP8).

5 Application Level

The Application level provides the interface between users and CONVERGENCE and is the top level of CONVERGENCE from the user point of view. Applications help users to perform tasks, such as creating, searching, browsing and downloading VDIs. This requires that applications should be integrated with the CONVERGENCE middleware. In this section, therefore, we will provide a brief overview of middleware interactions with tools and application. Details of specific applications are provided in deliverables from WP7.

In general, CONVERGENCE applications access CONVERGENCE through one of three mechanisms.

- For middleware operations that are performed remotely, they use Elementary Services. Typical examples of such operations include identification, packaging and delivery of content.
- To execute a pre-specified chain of Elementary Services, they use aggregated services.
- To call a chain of engines that execute locally, they use a client Orchestrator to monitor the status of each engine and synchronize individual procedures.

CONVERGENCE will support two kinds of applications: Java standalone applications and web applications. While the former can make direct web service calls to access an elementary or an aggregated service, or to execute a local orchestration, the latter require an HTTP proxy that performs the necessary operations. In detail, when a browser-based (web) application makes a request, the request is redirected to the proxy, which then decides whether it should make a CoNet call (e.g. when a middleware operation is requested) or a plain TCP/IP based call (e.g. when it is necessary to fetch a current web page). Whenever an application (e.g. the LMU podcast application or the FMSH video application) requires streaming it relies on CoNet. In this case, the HTTP proxy filters the client request and converts it to a call to CoNet. Finally, if the client side of the application (i.e. the browser) needs to make a local call to an orchestrator, the call is passed through the HTTP proxy, as described in detail in deliverable D7.1. [40]

The Application level is split into two sub-levels: a Tools sub-level and a User Applications sub-level (see Figure 28). The Tools sub-level contains functional components that combine a subset of CoMid functionalities and can be reused by many applications. Typical tools might include a VDI editor for creating new VDIs, a VDI subscription tool, a VDI browser tool and a VDI notification tool. In any given application, the Tools Layer is responsible for the interaction with CoMid; the Application level provides the interface to the user and visualizes search results in the format that best meets her requirements. For example, Photo VDIs or Video VDIs may appear differently in different applications.

By interacting with the CDS, tools can help the user to describe her resources. The CDS contains all the concepts available in the CONVERGENCE framework. Applications provide an interface to the service that users can exploit when they wish to describe a resource or define a set of search criteria.

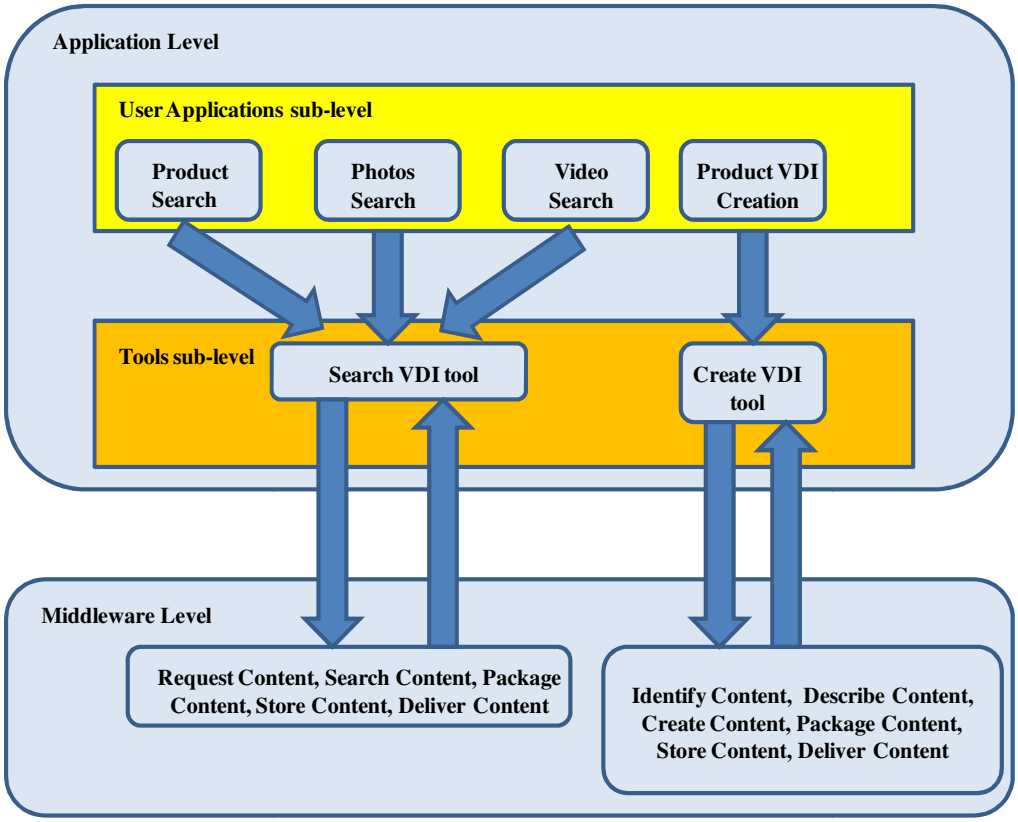


Figure 28 - Interaction of Tools and Applications sub-levels with CoMid

Each Tool contains a set of APIs that allow the development of new applications or the integration of existing ones. These APIs are responsible for parsing user inputs to CoMid and parsing CoMid outputs to the application GUIs. A detailed description of these APIs can be found in D7.2 ([45]). So far, we have identified the following Tools:

- User Registration: Allows the registration of a new user to a CONVERGENCE Application. The user fills in a form appropriate application-specific metadata for registration. Once the operation has been successfully completed, the tool returns a unique user identifier.
- Content Registration: This tool provides an identifier for a VDI or for a component of a VDI. The identifier allows authentication of the VDI on later occasions.

- Create Resource VDI: This tool allows a user to create an R-VDI, eliciting metadata to be includes in the resource as well as its URL and descriptors for links to other VDIs.
- Publish VDI: Once a user has created an R-VDI, she may wish to advertise it to CoMid. To do this, she uses a Publish VDI tool that copies relevant metadata from the original R-VDI.
- Revoke VDI: This tool allows a user to revoke a VDI from CoMid or CoNet.
- Subscribe to VDI: This tool allows a user to subscribe to specific VDIs by injecting subscription VDIs with appropriate metadata that describe her interests.

Additional APIs allow users to create licenses for the different types of VDIs, to create and parse metadata, and to create Event Reports.

Figure 29 provides a graphical representation of the interaction between these tools, application GUIs and CoMid.

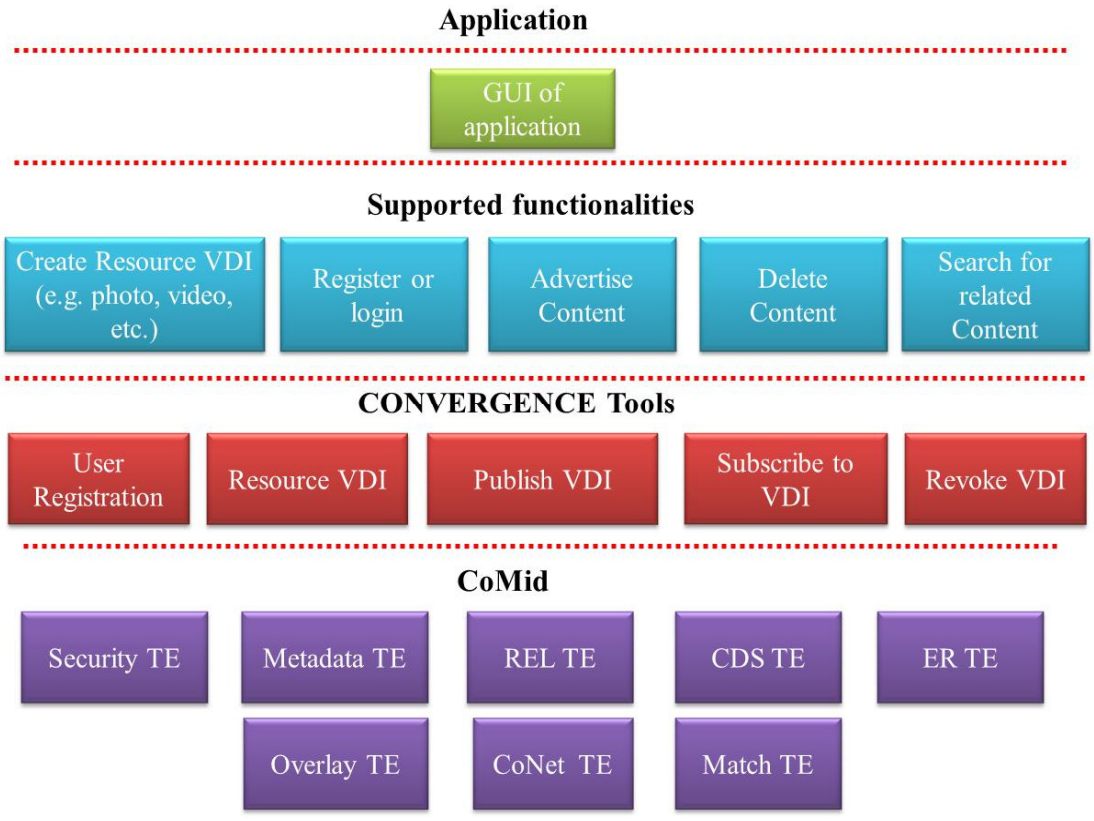


Fig. 29: A chain of Protocol and Technology Engines

6 Technical specification of CONVERGENCE Middleware (CoMid)

6.1 CoMid overview

The CoMid architecture is based on concepts developed in the MPEG-M (MPEG extensible Middleware) standard, an extension of the former MXM standard. CoMid functionality is implemented using MPEG-M Elementary Services [see working drafts of ISO/IEC 23006-4 at http://mpeg.chiariglione.org/working_documents.htm#MPEG-M], and, when MPEG-M does not provide the necessary functionality, by new CONVERGENCE-specific Elementary Services.

CoMid is based on a DDD (Describe, Discover, Distribute) paradigm. The middleware provides a set of standardized building blocks, APIs and protocols to describe, discover and distribute resources (VDIs) on the basis of "what" (metadata) they contain and offer. Higher-level Applications and Tools are built by assembling CoMid blocks into custom software, manipulated by users.

6.1.1 CoMid functionality

6.1.1.1 Describe functionality

Whenever a user produces a piece of multimedia content, or wants to market a good or service which is going to be governed and mediated through the CONVERGENCE system, the resource has to be described in a VDI. The VDI is a structured repository for all metadata and context information describing a specific resource. The CoMid "Describe" functionality is used to: i) access ontologies and directory services needed to tag resources; ii) create, identify and parse VDIs; iii) generate licensing information and contractual obligations, which accompany the resource.

6.1.1.2 Discover functionality

In large distributed networks, discovery of resources is crucial. CoMid supports semantic searches over the VDI space (by semantic search, we mean a structured and multi-criteria search over the semantic part of VDIs. The result is a ranked list of resources, matching the search criteria). The CoMid Discover functionality uses semantic criteria to search for VDIs and to manage the topology of the VDI space, making it possible to perform queries efficiently.

6.1.1.3 Distribute functionalities

Users interact with the CONVERGENCE system through an asynchronous, publish/subscribe paradigm. The CoMid Distribute functionality implements pub/sub abstraction end-points towards the user, supports unpublish operations and implements content-based subscription to resources.

6.1.2 Elementary Services, Protocol Engines and Technology Engines

The CoMid is organized around the concept of Elementary Services (ESs), implemented as Protocol Engines (PEs). Protocol Engines may call Technology Engines (TEs) to perform specific actions. PEs and TEs are assembled in a Service Oriented Architecture (SOA), based on the MPEG-M standard. Some of the Services needed by CoMid are directly available in the MPEG-M standard. Others will have to be specifically implemented within the CONVERGENCE framework. These services constitute a suite of building blocks facilitating the implementation of CONVERGENCE value chains (value chains are defined in MPEG-M as part of a business scenario). In such value chains, all devices are based on the same set of technologies, which they access via CoMid. These will include but not be limited to technologies standardized by MPEG. The use of a common set of technologies guarantees that all devices along the value chain can interoperate.

The section on Protocol Engines describes the set of PEs implementing ESs in CoMid and the way they map onto functional areas of the CONVERGENCE middleware.

CoMid also introduces the concept of Entities, meaning the objects on which ESs act. The main entities defined in CoMid are VDIs (equivalent to Content in MPEG-M nomenclature), Devices, Events, Groups, Licenses/Contracts, Services, and Users. By combining ESs and Entities, it is possible to describe a series of Operations - the basic building blocks in the CoMid SOA. These Operations define a corresponding set of protocols and APIs that enable any user of the CONVERGENCE system to access those services in an interoperable fashion.

6.2 Orchestration and Aggregation

In CONVERGENCE, applications communicate with CoMid either by calling a Protocol Engine, which processes a request and calls Technology Engines to fulfil the request, or by calling a Technology Engine directly (as may occur when the Technology Engine is running on the local client device). Figure 30 shows a typical example where an application calls three Protocol Engines, PE1, PE2 and PE3 which in turn call remote Technology Engines and another Technology Engine which runs locally on the client device.

Using CoMid may require various remote calls and coordination of the results (see again Figure 30). This means that the work of a CONVERGENCE application developer may be rather complex. The same applies to the Protocol Engine provider who will need to coordinate several Technology Engines.

CONVERGENCE will take advantage of the inherently clear organization of Protocol and Technology Engines to support automatic execution of chains of engines. This will facilitate the work of the applications/middleware developer. To this end, we introduce the *aggregator* and the *orchestrator*, two middleware components that coordinate the execution of the atomic operations defined by the protocol and the Technology Engines.

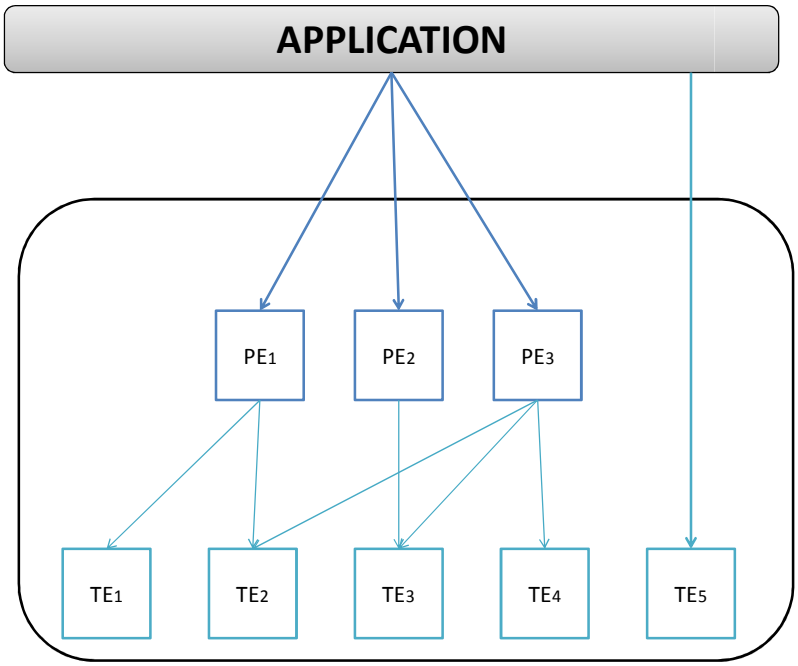


Figure 30 - Application Call to CoMid

Figure 31 illustrates how such an approach could be applied in action. Instead of calling each Protocol Engine separately, the application directly calls an *aggregation* of Protocol Engines, which exposes a new, (more) complex protocol, and performs the synchronization of the different engines.

Instead of calling and coordinating calls to various Technology Engines, the Protocol Engine relies on the *orchestrator*, which exposes a custom interface and relieves the Protocol Engine of the coordination of the calls. The orchestrator and the aggregator coordinate and execute a chain of engines transparently.

There are two ways of implementing the orchestrator and the aggregator on the CONVERGENCE platform. The first is to develop each aggregation/orchestration separately, defining the interfaces through which they can be called and programming them in the same way as any other engine. The second is to use workflow technology. More specifically we can represent chains of engines by BPMN (Business Process Model and Notation) workflows[6] . We can then use these workflows to dynamically create aggregations or orchestrations of engines. Given that workflows can incorporate executable processes, we can create BPMN processes, which automatically make the necessary calls to the Protocol and the Technology Engines, from within a general workflow environment.

[6] MPEG-M Part 5 [32] defines the workflow for the chain of Protocol Engines involved in Service Aggregation

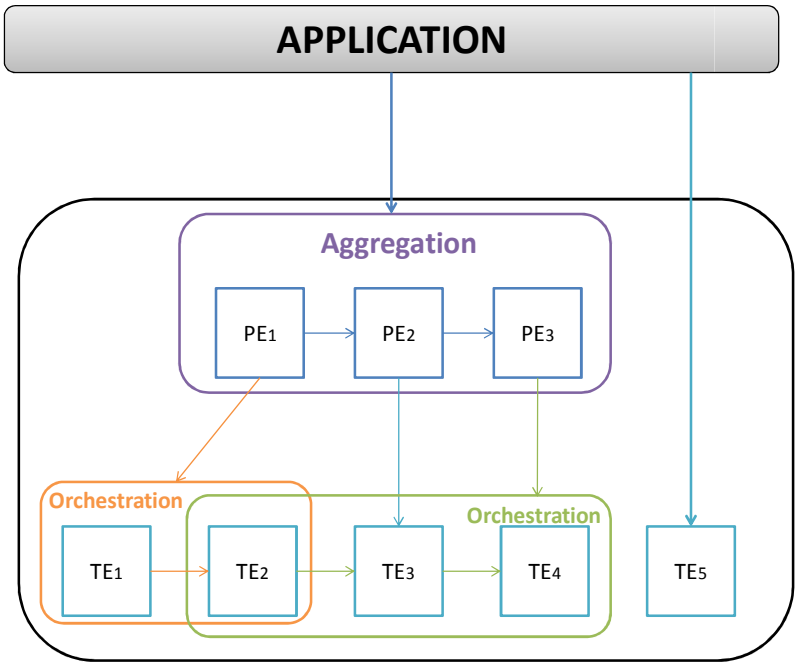


Figure 31 - Aggregation and Orchestration in CoMid

At the time of writing, we are considering both orchestration solutions. On the one hand, we are examining APIs for state of the art workflow engines (the second solution). On the other we are studying BPMN v2.0 as it is used to support service aggregation in MPEG-M Part 5 (the first solution). As part of this work, we have designed a general workflow environment that takes a BPMN v2.0 executable process as input (using a script task for each call to a method exposed by the Technology Engine), executes the process and returns the result to the caller.

6.3 Protocol engines

The table below lists all CONVERGENCE Protocol Engines. Engines extended by CONVERGENCE are shown in *italics*, engines specified by CONVERGENCE are shown **in bold**.

Operation	***Entity***	***Definition***
Authenticate	Content	Allows a user to check the authenticity of a VDI on his/her device, by using the VDI itself and its digital signature. If the digital signature is not embedded inside the VDI, it has to be provided separately.
	Device	**Enables a user or a client application to authenticate the device where it is running. This way, the owner of the content used by the application can be sure that the device is trusted.**

	User	Allows the client to confirm a user's identity. This service will work with users belonging to different domains using a Single-Sign-On (SSO) mechanism.
Create	Content	Enables the creation of a Versatile Digital Item, by providing the data it contains, such as resources, descriptions, licenses etc.
	License	Enables the creation of a license, by providing the corresponding information expressed in a machine readable format. CONVERGENCE will support the MPEG-21 Rights Expression Language (REL) along with possible extensions to be defined in D4.2.
Deliver	Content	Enables the client to request the delivery of a specified VDI according to terms and conditions, specified in a license. The Deliver Content Service Provider does not necessarily have to be the sender or the receiver of the content; it is just responsible for performing the transaction between the two parties
Describe	Content	Enables a user to generate, provide and retrieve descriptions of VDI content. The service provides 1. A protocol for setting the description of a VDI 2. A protocol for getting the description of a VDI **3. A protocol that facilitates the generation of semantic descriptions of VDI contents by providing relevant entities from ontology models** The Search Content Elementary Service uses these descriptions to perform search.
Identify	Content	Enables a User to obtain an identifier for a VDI or any of its components.
	User	**Enables a User to obtain an identifier. The identifier is stored by the Service Provider, so that an application can use the user's credentials to retrieve it at a later date**
Inject	**Content**	**Makes a VDI discoverable via semantic search operations. This operation can be performed by users or automatically by the system, whenever a new VDI is published.**
Package	Content	Prepares a VDI for delivery, viz. creates a file representing the VDI or a stream of VDI fragments and binds it to the transport protocol
Present	**Content**	**Enables a user to set a style for the presentation of a VDI. The Service Provider will apply the style to the VDI and return the corresponding formatted document.**
Process	Content	Enables a user to modify the contents of a VDI. The result will be a new VDI, that complies with the semantics of the update VDI and maintains the same sequence identifier as its predecessor.
	License	Enables a user to create a new license, based on an existing license. The new license can refer to new resources, and principals or be based on its predecessor.
Request	Content	Enables a user to access a VDI.
Request	Event	Enables a user to ask for an Event Report Request associated with a given VDI, or the Event Reports associated with a given Event Report Request.
Revoke	**Content**	**Enables a user to unpublish/unsubscribe VDIs**
Search	Content	Enables a user to search for VDIs published in CONVERGENCE. The query may use the following classes of search criteria:

		• Unstructured criteria: free text description. • **Semi-structured criteria: property-value pairs where the property is based on a model and value is free text.** • **Structured criteria: property-value pairs where the property and the value are both based on a model.** In the last two cases, the user can use the Describe Content Elementary Service to request entities belonging to existing models and construct the criteria for the query.
Store	Content	Allows the transfer and the storage of a VDI in a local or remote device.
	Event	Enables a user to store an Event Report Request or an Event Report that may occur on a device.

6.4 Technology engines

The table below lists all CONVERGENCE Technology Engines. Engines extended by CONVERGENCE are shown in *italics*, engines specified by CONVERGENCE are shown **in bold**. A complete description of these engines can be found in deliverable D5.2.

Technology Engine	***Definition***
CDS TE	The Community Dictionary Service Technology Engine (CDS TE) provides semantic services to the CONVERGENCE middleware
CoNet TE	The CoNet Technology Engine provides the CONVERGENCE Middleware with access to the features provided by CoNet, Information-Centric Networking
Digital Item TE	The DI Technology Engine interface defines methods for operating on ISO/IEC 21000-2 Digital Item Declaration (DID) data structures
Event Report TE	The Event Report Technology Engine (TE) defines methods for operating over ISO/IEC 21000-15 Event Reporting data structures
Media Framework TE	The Media Framework TE is a high level engine, grouping several media specific engines (e.g. Video, Image, Audio and Graphics Engines). Each exposes APIs to create (encode) and access (decode) their respective elementary streams. The Media Framework TE also implements common functionalities (independent of media type) such as resource loading and saving
Match TE	The Match Technology Engine (TE) is responsible for matching subscriptions and publications. To support the CONVERGENCE content-based publish/subscribe paradigm, matches are based on descriptions of content, as detailed in previous sections. The Match TE identifies Semantic descriptors within the VDI structure and fires the corresponding parser.
MPEG-21 File TE	The MPEG21 File TE interface defines methods for operating on MPEG-21 File Format files
Overlay TE	The Overlay TE is responsible for distributing content to a set of peers,

	which are interested in that type of content. The CONVERGENCE publish-subscribe mechanism uses this TE to broadcast publication and subscription VDIs to a group of peers that have already published or subscribed to content of the same type.
REL TE	The REL Engine interface defines methods for operating over Rights Expression Language (REL) data structures, as defined by ISO/IEC 21000-5.
Security TE	The Security TE interface defines security-related methods.

7 The Computing Platform

7.1 Network component

The CONVERGENCE Network (CoNet) provides Network level functionalities. CoNet is an information - Centric inter-network that provides users with a network access to remote named-resources, rather than to remote hosts. Named-resources can be either data[7] ("named-data") or service-access-points[8] ("named-service-access-points"), identified by a network-identifier (a name).

The name of a resource is its *network-identifier* (NID). Unlike Internet URLs, which include information about "where" a resource is located, CONVERGENCE NIDs do not necessarily contain a reference to a resource's location. A network-identifier is an *anycast* address: a system may contain multiple replicas of the same named-resource. Network functionality will connect a user to the "best" (e.g. the closest) replica. For example:

- A PDF copy of the TIMES newspaper for Sept-30-2010 could have the name "the times: Sept-30-2010";
- The service-access-point for an SQL database service provided by the Foo company could have the name "Foo: sql-database";
- The service-access-point for an MXM Content-provider device owned by the Foo company could have the name "Foo: mxm-content-provider".

As it handles named-resources through their network identifiers rather than IP addresses, the network infrastructure is aware of the resources it is handling. This awareness can be exploited to support anycast routing, resource replication, and in-network caching (if resources are data and not points of access to a service). Current IP networks do not provide this very useful functionality[9]. The possibility of accessing resources without having to specify their location, simplifies mobility issues. It also means that resource names can be portable and do not depend on the Service Provider providing the resource.

The CoNet allows users (e.g., end-users and providers) to advertise and revoke their named-resources. A resource can be replicated in different geographical locations by adopting the same network-identifier. Replication enables users to exploit resources that are close to their location and improves reliability of access. The CoNet provides *secure distribution* of resources by ensuring that the network-identifier

[7] Named-Data include: documents, video, images, structured information, VDIs. The Network level is a general purpose one, as it can handle any kind of data and not only data generated by the CONVERGENCE system.

[8] A named-service-access-point is a network endpoint through which an upper layer entity (e.g., a server or a client) sends and receives data. In the actual Internet, for instance, TCP port n. 80 is the default service-access-point for HTTP servers.

[9] Some partial support is provided by proprietary systems, such as Content Delivery Networks (e.g. Akamai).

cannot be forged and that receivers can verify the validity and provenance of named-resources [2][5]. Network functionalities can autonomously cache transiting resources [4], if this is allowed by security rules and by the nature of the resource[10].

When a named resource is named-data, the CoNet provides the means to deliver it to intended recipients. It can also facilitate the delivery of the named-data through replication/caching.

When a named resource is a named-service-access-point, the CoNet provides the means to exchange data between a requesting upper layer entity and the upper layer entity addressed by the named-service-access-point. This functionality is needed by "traditional" client/server services (e.g. HTTP, POP, SMTP) and middleware services (e.g. transport of middleware service data), which do not assign names to their data. To support this need, we introduce the concept of "*un-named-data*", i.e. data not identified by a NID. The named-sap case can be extended to multicast; in this case the NID of a named-sap should be interpreted in the multicast rather than the anycast sense.

The CoNet thus offers two mechanisms for the delivery of data.

1. When users need to access named-resources that are data (e.g., documents, VDIs, files, etc.), the CoNet allows them to download actual content. In this case, the user asks the network to provide a resource with a given network-identifier and the network provides her with the actual content, without invoking any other upper layer functionality. This *modus operandi* follows the *Information-Centric networking* paradigm recently proposed in the literature.
2. When it is necessary to support service sessions for upper layer entities (e.g. a client-server couple), the CoNet couples the local upper layer entity with the named-service-access-point for the "best" remote upper layer entity providing the service (via anycast routing), going on to support an interactive exchange of un-named-data between these two upper layer entities. In the case of a client-server service session, for example, the un-named-data is upper layer data (e.g. HTTP, SMTP, POP, SQL, MXM messages) exchanged between a local client and a remote server (e.g. an HTTP server, a middleware device etc.). Thanks to this functionality, the CoNet can natively support most current Internet services, existing middleware protocols and any service that requires point-to-point bidirectional interaction or point-to-multipoint communications. This *extension* to the capabilities of a "plain" Information-Centric network [3] allows the CoNet to support not only content retrieval but also more traditional services.

Access to a named-resource involves two steps: i) anycast routing of a user "request" to the best CoNet node providing the named-resource and ii) unicast routing of the "response" from the selected CoNet node toward the requesting CoNet user. Anycast routing uses the network-identifier of the named-resource, as previously discussed. In unicast routing, endpoints are identified by a *location-identifier*

[10]It is possible to cache data but not points of access to a service.

(LID) – the equivalent of the IP address and port number on the current Internet. The use of location-identifiers makes it possible to support sessions that require interactive exchange of data between a "client" and a "server". After the first phase of anycast routing, when the CoNet selects the best server, further communications between the client and the server can be routed using their location-identifiers; in fact, if the client application tried to reuse the (anycast) network-identifier, the CoNet might select a different server, interrupting the session[11.12].

7.1.1 CoNet Architecture

The CoNet is an (inter-)network layer that provides users with a network access to remote named-resources. The main features of CoNet can be summarized as follows.

- It is stateless: network nodes do not maintain information on the ongoing communications.
- It limits the size of name-based routing tables by caching only a subset of all possible routes; missing routing entries are looked up in a name-system and then cached.
- It can be integrated in existing IP networks by using a new header option, which makes IP content-aware [6]. In this case, the nodes could use hybrid routing tables containing both IP network addresses and names. However, CoNet also supports the clean-slate and overlay deployment approaches.

CoNet is a system that interconnects CoNet Sub Systems (CSSs) (see Figure 32). A CSS contains CoNet nodes and exploits an under-CoNet technology to transfer data among CoNet nodes. A CSS could be:

- A couple of nodes interconnected by a point-to-point link, e.g. a PPP link or a UDP/IP overlay link.
- A layer-2 network, e.g. Ethernet, or a layer-3 network, e.g. a private/public IPv4 or IPv6 network, or a whole IP Autonomous System, or even the whole current Internet.

[11]One way to implement the location-identifier (without exploiting IP) would be to adopt a *link-layer source-routing* approach. In this case, the structure of the location-identifier would represent the sequence of link-layer interfaces to be followed from the CoNet node (Server) to the requesting CoNet user, i.e. on the reverse path. This sequence is built during the preceding anycast routing phase. The location-identifier has a *temporary* meaning; it is assigned in a distributed way, and has no impact on the routing-plane of the CoNet, which is concerned only with network-identifiers. *Link-layer source-routing* is the approach adopted by the PSIRP FP7 project (http://www.psirp.org/).

[12] It has not yet been decided whether the CONVERGENCE system will support also service sessions between two NIDs, when it happens that NIDs are unicast addresses and thus univocally identifies a service access point.

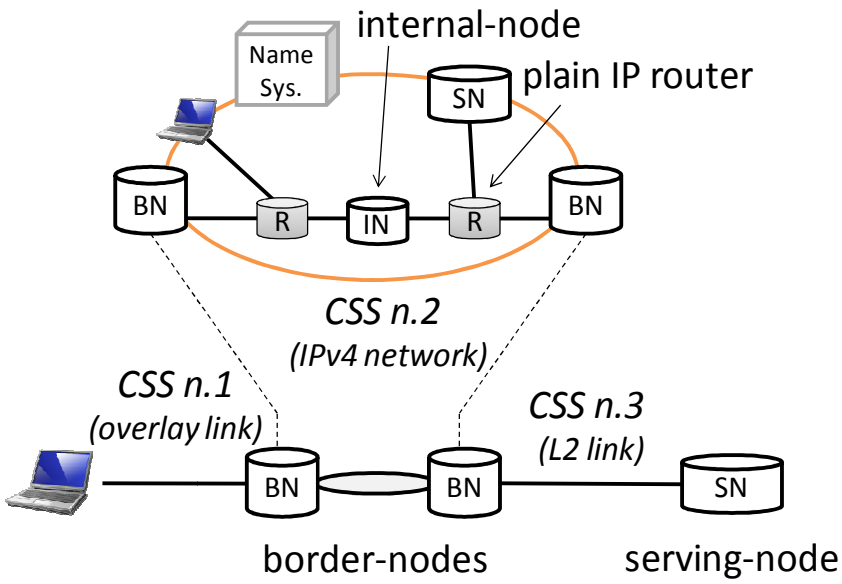


Figure 32: CoNet Architecture

The devices within a CSS use an autonomous and under-CoNet homogeneous address space and, if necessary, an interior under-CoNet routing protocol (e.g. [13]).

CSSs can be defined rather freely. For instance, if CoNet protocols are implemented only in user equipment, interconnected by the current Internet, we have only one CSS: the current Internet. If CoNet protocols are implemented in current border gateways (i.e. gateways running BGP), CSSs coincide with current Autonomous Systems. If CoNet protocols are implemented in all current routers, then CSSs coincide with current IP subnets. If CoNet protocols are implemented in nodes that interconnect different layer 2 networks, removing IP, CSSs coincide with these layer 2 networks.

CoNet nodes exchange *CoNet Information Units (CIUs)*: *interest CIUs* convey requests for named-data; *named-data CIUs* transport chunks of named-data, e.g., parts of a file (see Figure 33). To best fit the transfer units of an under-CoNet technology, all CIUs are carried in smaller CoNet data units named *carrier-packets*.

CoNet nodes are logically classified as end-nodes (ENs), serving-nodes (SNs), border-nodes (BNs), internal-nodes (INs) or name-system-nodes (NSNs). *End-nodes* are user devices that request named-data by issuing interest CIUs. *Serving-nodes* store, advertise and provide named-data by splitting the related sequence of bytes into one or more named-data CIUs, which are transferred by means of carrier-packets (see Figure 33). *Border-nodes,* located at the border between CSSs, forward carrier-packets by using CoNet routing mechanisms (i.e. routing-by-name and inter-CSS source-routing, as described below) and cache named-data CIUs. Optional *Internal-Nodes* can be deployed *inside* a CSS to provide in-network caches. Unlike border-nodes, internal-nodes only use under-CoNet routing mechanisms to forward

carrier-packets. The CSS uses optional *Name-System-Nodes* to assist the CoNet routing-by-name process.

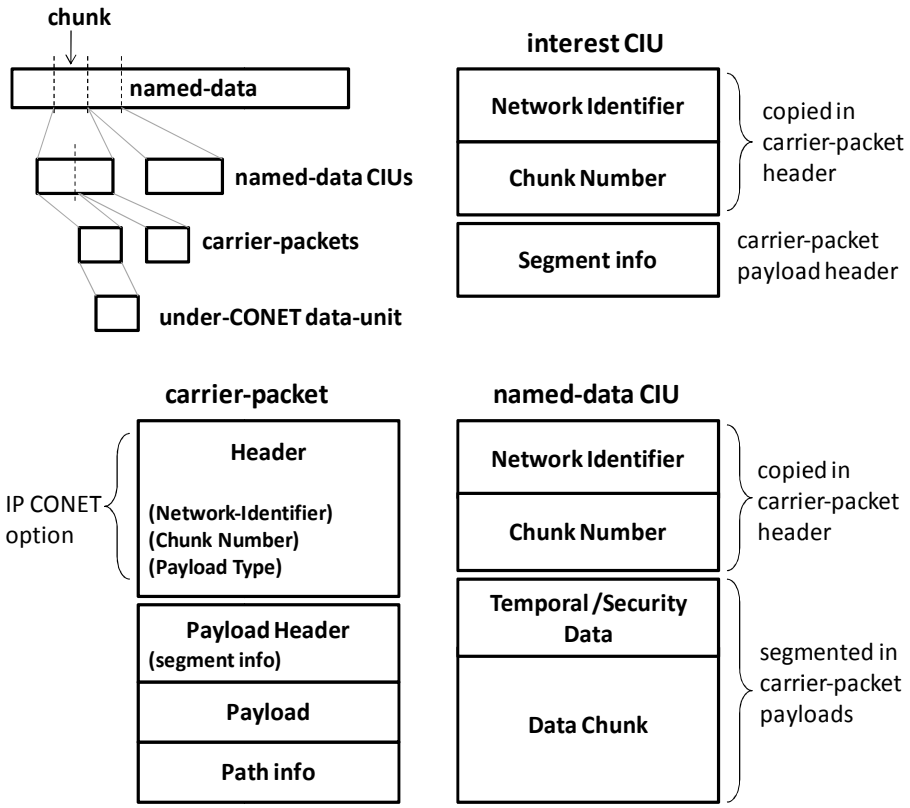


Figure 33: CoNet Information Units (CIUs) and carrier-packets

There are three possible approaches to deploying CoNet:

- *The overlay approach*: CoNet runs on top of the IP layer; CSSs are couples of nodes connected by overlay links, e.g. UDP/IP tunnels, as in CSS n.1 in Figure 32;
- *The clean slate approach*: CoNet runs on top of layer-2 technologies (e.g. Ethernet, PPP, MPLS LSP); CSSs are nodes connected by layer-2 links/networks, and CoNet replaces the IP layer, as in CSS n.3 in Figure 32;
- *The integration approach*: CoNet functionality is integrated in the IP layer by means of a novel IPv4 option [6] or by means of an IPv6 extension header, as in CSS n.2 in Figure 32.

The three approaches are not mutually exclusive, but can be combined. Variants of the clean-slate and overlay approaches have already been discussed in the literature and by other research projects [7], [8], [9]. However, to the knowledge of the authors, the proposed integration approach is novel.

7.1.2 Model of operations

In this section, we describe the operation of CoNet in the scenario depicted in Figure 32. In this scenario, an end-node retrieves a chunk of a named-resource from a serving-node. We assume that the serving-node has already advertised the related network-identifier in the CoNet, using a name-based routing protocol, as described in section 7.1.4. The retrieval of named-data thus involves a sequence of *request - delivery* phases in which the end-node requests and obtains named-data CIUs and reassembles the whole named-data (Figure 33). For simplicity, we consider a case in which the named-data is fully contained in a single named-data CIU that, in turn, is fully contained in a single carrier-packet. This means we only need a single request-delivery phase.

Request

- An end-node requests the named-data CIU by issuing an interest CIU, which includes the network-identifier for the named-data; the interest CIU is encapsulated in a carrier-packet, named *I*.
- The end-node and intermediate border-nodes *route-by-name* the packet *I*. The route-by-name process singles out the *CSSaddress*[13] of the next border-node towards the serving-node, on the basis of the network-identifier contained in *I*. Then, the routing engine encapsulates the carrier-packet *I* in the under-CONET data-unit and uses the CSS address as the destination address.
- The CSS address of the end-node and the set of CSS addresses of border-nodes in the "upward" path are appended to the carrier-packet *I*, in a control field named *path-info*[14];
- The internal-nodes parse carrier-packet *I* and forward it, using the under-CONET routing engine.

Delivery

- The first in-path CONET node (BN, IN or SN), which is able to provide the named-data CIU requested by *I*, sends back the CIU, without propagating the request further.

[13] A CSS address is an address consistent with the under-CONET technology traversed by the packet (e.g. an IPv4 address).

[14] This info will be used to find the reverse-path to route the named-data CIU back to the requesting node, in the delivery phase. In [7] the same goal is achieved by maintaining states in network nodes. We propose to use source-routing. We are aware of the trade-offs involved when the number of CONET border nodes traversed is limited (i.e. CSSs coincide with Internet Autonomous Systems). As an alternative, the path-info field could contain the NID of a named-sap, specifying where the requesting end-node can be reached. In this case, reverse-path routing could be performed by means of route-by-name procedures. This would be more convenient if CSSs were smaller and the number of traversed CONET border-nodes higher. It would also give the network operator more freedom to choose the reverse-path.

- The named-data CIU is encapsulated in a carrier-packet, named *C*. The carrier-packet *C* traverses the same CSSs as the carrier-packet *I*, but in the downward direction, until it reaches the requesting end-node.
- The serving and the border nodes perform *inter-CSS* reverse-path routing in a *source-routing* fashion, using the path-info control field. This path-info is the copy of the path-info set up in *I* during upward routing.
- Within a CSS, the under-CONET technology (e.g. IP) routes carrier-packet *C*. In these conditions, we can use traditional traffic engineering mechanisms.
- All border-nodes and internal-nodes in the downward path can cache the named-data CIU contained in *C*.

We observe that the use of inter-CSS source-routing on the reverse-path does not require "pending" states in the nodes traversed by the packet. In end-to-end sessions in the same IPv4 CSS, the path-info field is not necessary, as its only content would be the IP address of the end-node, already contained in the IP header.

7.1.3 CoNet protocol stack

As shown in Figure 34, every CoNet node has the CoNet and the *Under-CoNet* layers. The CoNet layer is connectionless, handles CIUs and carriers-packets, and provides other functionality (e.g. caching, security, etc.).

The end-nodes also have transport-level functionality, support reliability and flow control, and provide an Applications Programming Interface (API) (see deliverable 5.2) for the definition of the API between CoNet and upper layers). In what follows, we adopt the receiver-driven TCP-like approach proposed in [7], adapting it to our own terminology.

In this approach, the transport algorithm issues a sequence of interest CIUs and each of them requests only a small part of a named-data CIU, e.g. 1500 bytes per interest CIU. By controlling the sending rate of these interest CIUs, it is possible to obtain a TCP-like flow control mechanism. For instance, we could replace current TCP ACKs with interest CIUs and apply TCP congestion-window concepts to in-flight interest CIUs.

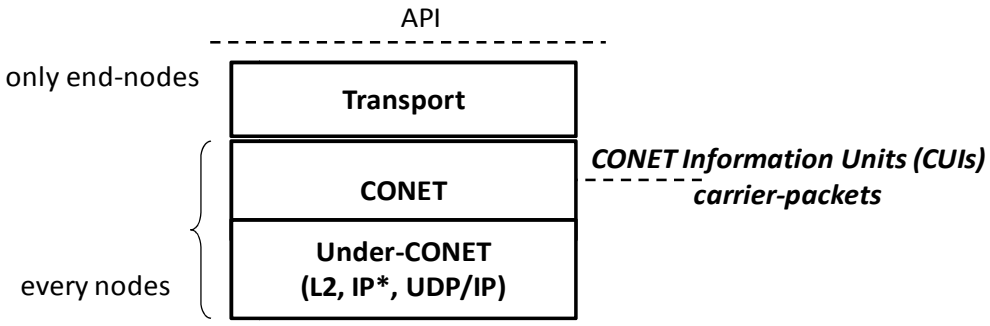


Figure 34: CoNet Protocol Stack

Figure 33 shows the packetization process for the CoNet CIUs (interest and named-data) and carrier-packets. In terms of notation, our interest CIU and named-data CIU correspond to the "interest packets" and "data packets" proposed in [7]. However the protocol information is different. For example, our interest CIUs carry the segment info field which is not present in [7]). In addition, we introduce the concept of carrier-packets, designed to improve CoNet forwarding speed.

Details of CoNet protocol operations are provided in deliverable D5.1 [10], which also describes the solution used when the CSS is an IP network, e.g. the IPv4 network in Figure 32.

The named-data (i.e. the representation in bytes of a resource) is split into different chunks. The optimal chunk size is the result of several tradeoffs. We favour a size roughly equivalent to the size of chunks in current P2P systems, e.g. 256-512 kbytes. However, the CoNet architecture can support variable chunk sizes.

Each chunk is inserted into a named-data CIU, the data-units for the caching process. The control information includes the network-identifier and the chunk number, as well as timing and security data.

The network-identifier is a tuple <*namespace ID*, *name*>. The *namespace ID* determines the format of the *name* field. The *name* field is a namespace-specific string. Each namespace uses its own rules to generate unique *names* with its own format. We specify a default naming format, where the *name* is the composition of two hash values, i.e. *name*=<*hash (Principal identifier)*, *hash (Label)*>. Principal identifier and label are flat-names and a hash function transforms them to a fixed number of bytes (e.g., 6 bytes). A principal is the owner of her named-data and uses a *Principal identifier* whose hash is unique in the namespace. *Label* is an identifier chosen by the principal to uniquely differentiate her named-data. For instance, we could define the namespace "www" for the names of web resources (defined using current naming conventions), use the domain name (e.g. www.cnn.com) as the principal identifier and the URL (e.g. /foo/index.html) as the label.

Timing-data includes time information, like the expiry date, which can be exploited to implement digital forgetting mechanisms. Security-data make it possible to validate a named-data CIU before caching it or delivering it.

An interest CIU is a request for a set of bytes belonging to a named-data CIU, e.g., all bytes from byte X to byte Y (segment info field) of the named-data CIU n. Z (chunk number field).

Carrier-packets are low-level carriers of CIUs and are the data-units of the forwarding process. Carrier-packets are reassembled in border-nodes or in internal-nodes that want to cache the related named-data CIU, and in end-nodes; this operation is necessary to validate the content. Given that a named-data CIU could be too large to be transported by a single under-CoNet data-unit (e.g. 1.5kB for Ethernet and 64kB for IP) we introduce the concept of carrier-packets. Carrier-packets, tightly associated with specific communication sessions between an end-node and a serving-node or a cache, make it possible to perform source-routing.

7.1.4 Name-based routing: lookup-and-cache

Name-based routing is the mechanism used to update CoNet name-based routing tables. These are used by end-nodes or border-nodes to route-by-name interest CIUs. An entry in the name-based routing table contains the tuple <network-identifier, mask, next-hop, output-interface>. This is like an IP routing table entry. However, instead of net-prefixes we have *name-prefixes*, i.e. <network-identifier, mask> couples. Next-hop is the *CSS address* of the next border-node toward the serving-node.

In [11][7] the authors propose the use of traditional routing protocols, e.g. BGP or OSPF, to disseminate name-prefixes. We call this approach *prefix-dissemination*. The disadvantage of this system is that aggregation of names (i.e., network-identifiers) is not efficient unless names include information about the location of the serving node [12]. As a result, *prefix-dissemination* is likely to produce large name-based routing tables. As an example, consider how it might be possible to support DNS domain names (as we would like to). In principle, it could be possible to perform limited location-based aggregation, using domains [7]. However, this would not work for generic top-level domains (.com, .net, etc.), which include very large numbers of names (the .com domain currently has about 90 million) in many different locations. Even in the case of country-code top-level domains aggregation is unlikely to be very efficient. For instance, about 30% of .it names are outside Italy. Given that it is not feasible to include all possible names in the routing table, we propose a name-based routing scheme that we call *lookup-and-cache*. In this approach, a CoNet node (end-node or border-node) uses a fixed number of rows in its name-based routing table as a *route cache*. When a node misses the routing info required to route-by-name an interest CIU, it looks up its routing entry in a DNS-like *name-system* and inserts the new entry in the route cache. When all rows are full, new routing entries replace old ones according to a suitable policy. From a logical point of view, a name-system serves a single CSS and a single namespace.

If a serving-node belongs to the same CSS as the node requesting the routing info, the name-system returns the CSS-address of the serving-node. If the serving-node is outside that CSS, the name-system returns the CSS-address of the egress border-node. If there are more than one serving-nodes, or egress border-nodes (due to replication operations), the name-system uses known techniques [14] to select the most convenient destination.

Prefix-dissemination and lookup-and-cache approaches can work separately or they can be combined, e.g. by using prefix-dissemination for the most popular named-data and lookup-and-cache for less popular data.

7.1.5 Integrating CoNet in IP

In this section, we describe a technique to support the CoNet in a CSS that is an IP network (IP-CSS), e.g. CSS n.2 in Figure 32. The IP network could correspond to the whole public Internet. Our approach thus provides a way to offer CoNet services over the Internet. To achieve this goal, we propose what we call an *integration approach.* This approach i) does not require us to give up IP, as in the clean-state approach; ii) performs better than running CoNet on top of IP, as in the overlay approach. The idea of the integration approach is to *make IP content-aware*. To this end, we propose to transport the header of a CoNet carrier-packet in a novel IPv4 option (or IPv6 extension header) that we call the CoNet option (see Figure 33 and [6]). In this kind of IP-CSS, border and internal CoNet nodes are simply IP routers extended with CoNet functionality.

Figure 35 shows a possible architecture for a border or internal CoNet node. A *fast forwarding path* handles forwarding operations for CoNet carrier-packets and for plain IP packets. The hardware (RIB or FIB) routing table does not only include IP net-prefixes as entries but also name-prefixes. A name-prefix entry may refer either to remote named-data or to local cached named-data. In the latter case, the routing entry points to the local cache engine. Whenever the CoNet node receives an Interest CIU that matches the routing entry, it will return the locally cached named-data. Other CoNet and IP functions with less stringent delay constraints are performed by a CPU. Examples include IP and name-based routing, execution of caching algorithms, reassembly of named-data CIUs for caching, replies to interest CIUs requesting cached data, etc.

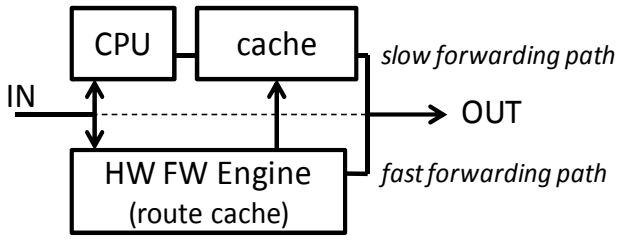


Figure 35: Architecture of a CoNet node of an IP-CSS

Most of these operations require parsing of incoming CoNet CIUs. With our approach, it is possible to forward an incoming CIU using the HW engine (without affecting forwarding performance) and in parallel to process it in the CPU in the so called “slow-path”.

The advantages of this approach with respect to the overlay approach is that it allows CoNet nodes to quickly forward carrier-packets, without having to perform slow deep packet inspection. This is a fundamental requirement for the implementation of Information-Centric features in nodes with high packet rates requiring fast forwarding. Additionally, the integration approach makes it possible to restrict deployment of CoNet routing-by-name functions to a subset of nodes (i.e. border-nodes and end-nodes) while allowing caching in all nodes running the new IP option (i.e. internal nodes). By contrast, in the overlay approach, the only way to implement caching in all nodes is to deploy routing-by-name functionality in all nodes.

The disadvantage of the integration approach is that it requires a new IP option. However this is far less disruptive than the clean-state approach. The approach lends itself to different deployment scenarios. In an extreme case, the IP-CSS could be the whole Internet and routing-by-name functions would be performed only in end-nodes. All that would be required to implement in-network caching would be to introduce the new IP option; there would be no need to introduce routing-by-name functions in routers.

In another scenario, we could partition the Internet into a set of IPv4 CSSs that interoperate exclusively through CoNet protocols. In this scenario, each CSS would uses an IPv4 addressing scheme unique to the CSS. IP routing would thus be restricted to individual CSS. This would allow new providers to offer public CoNet services without using public IP addresses.

7.1.6 CoNet Application Program Interface

This section describes the CoNet Application Programming Interface provided by the CoNet. ***Table 7*** reports a preliminary set of primitives. Thanks to these primitives, the CoNet allows its users to:

- Advertise named-resources, i.e. resources identified by a network-identifier (NID)
- Access named-resources
- Deliver named-data and un-named-data (i.e., upper layer data).

The scope of applicability of some primitives depends on the type of the named-resource they deal with. We thus distinguish two classes of named-resources: named-data (i.e., documents, video, images, structured information, VDIs, etc.) and named-service-access-points.

For examples of the use of these primitives see the next section.

Table 7 – Service primitives offered by the CoNet API

Service Primitive	Applicability	Origin	Output	Description
Advertise (NID, resource-type, [resource /port], expiry, auth,…)	Named-data and named-service-access-points	CoNet user (e.g., application or middleware)	success /failure	This primitive is used by a CoNet user to make a local named-resource accessible to other CoNet users. NID is the network-identifier of the named-resource. resource-type specifies if the resource is named-data or a named-service-access-point. In the case of named-data, resource is actual content (e.g., a document, an image, media, file, a VDI). In the case of a named-service-access-point, port is a *local entity port* where an entity expects to receive data through an Indication primitive (this corresponds to the transport port that IP networks use as a local address in a host to deliver data to a given entity). expiry defines a date after which the advertisement will be automatically revoked. auth is a set of information that authenticates the user advertising a named-resource with a specific NID. This prevents the forging of NIDs. Failures may occur if Auth is not valid, a resource type is not consistent, etc. After Advertise, a new named resource is introduced into the network. The resource will be reachable by name by any entity in the CONVERGENCE network. Given that CoNet supports routes-by-NID, this information has to be introduced into the network to route requests for the new named resource towards the appropriate network nodes Thus, Advertise changes routing information on CoNet.
Update (NID, resource-type, resource/port, expiry, auth,…)	Named-data and named-service-access-points	CoNet user (e.g., application or middleware)	success /failure	This primitive is used by a CoNet user to update the resource associated with a previously advertised NID. The meaning of the parameters is the same as in Advertise. However, in this case it is not necessary to introduce or modify routing information.

Revoke (NID, auth,…)	Named-data and named-service-access-points	CoNet user	success /failure	This primitive is used by a CoNet user to revoke (and delete) a local named-resource, identified by NID. auth is a set of information that authenticates the user seeking to revoke the named-resource. A failure occurs when the named-resource is not available. Like Advertise ,Revoke modifies CoNet routing information
Get (NID)	Named-Data	CoNet user	data / failure	This primitive is used by a CoNet user to retrieve a named-resource identified by a NID. A failure occurs if the named-resource is not available.
Send2Name (NID, data, port)	Named-service-access-points	CoNet user	success / failure	This primitive is used by a CoNet user to send un-named-data (i.e. upper layer data) toward a named-service-access-point, identified by a NID. If the CoNet user waits for a response from the remote endpoint, the parameter port is the local entity port where the CoNet user expects the Indication primitive.
Indication (LID, data)	Service access point	CoNet		When un-named-data (i.e. upper layer data) are received by a service-access-point, this primitive is invoked by the CoNet on the entity which is listening to the local entity port associated with the service-access-point (see advertise). The primitive gives the un-named-data and the temporary identifier LID of the location of the remote entity that has sent these un-named-data to the service entity listening to the port.
Send2Location (LID, data, port)	Service access point	CoNet user		This primitive is used by a CoNet user to send un-named-data (i.e. upper layer data), via unicast, to a specific entity whose location is temporary identified by LID. If the CoNet user expects a response from the remote endpoint, the parameter port is the local entity port where the CoNet user expects the Indication primitive.

7.1.7 Examples of use of the CoNet Application Program Interface

In this section, we show how the CoNet API can be exploited to support typical operations. Details are preliminary. The final specifications will be completed at a later stage in our work.

7.1.7.1 Advertising local data

Figure 36 depicts the case of a CoNet user (e.g. a provider application) who wishes to make the local file "times09-30-10.pdf", available to other CoNet users. The file has network-identifier (NID) "the times: Sept-30-2010". The provider calls **Advertise**. The local CoNet node stores the file locally, updates an internal database that maps "the times: Sept-30-2010" to the file "times09-30-10.pdf," and updates the routing plane of the whole CoNet, so that the local CoNet node becomes a provider of the named-resource "the times: Sept-30-2010". There may also be other providers.

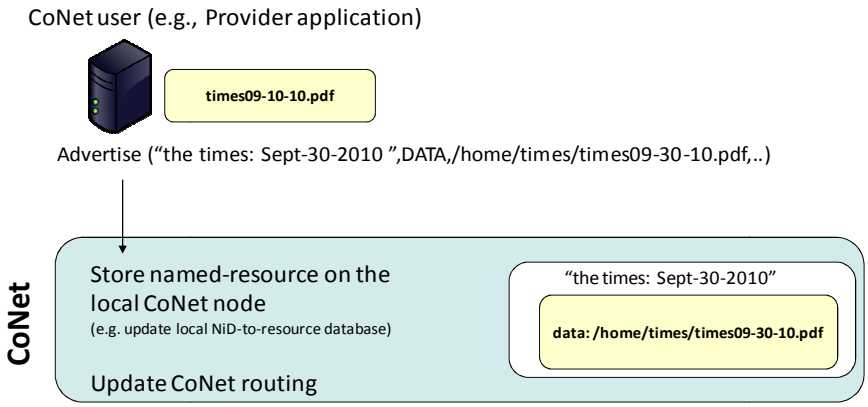


Figure 36: Advertising a local named-resource (named-data)

7.1.7.2 Advertising local services

Figure 37 depicts the case of a CoNet user (e.g. a provider application) who wishes to make a local SQL database service available to other CoNet users. The service has network-identifier (NID) "Foo: sql-database". The provider runs the relevant server application, which is configured to receive its service-data (un-named-data for the CoNet) through the local entity port 0x067497. Then the provider application uses the Advertise primitive. Subsequently, the local CoNet node updates an internal database that maps "Foo: sql-database" to the local entity port 0x067497 and updates the routing plane of the whole CoNet, so that the local CoNet node becomes a provider of the named-resource "Foo: sql-database". To improve the reliability of database access, the Foo company may replicate the database by repeating the same advertising operation on different CoNet nodes.

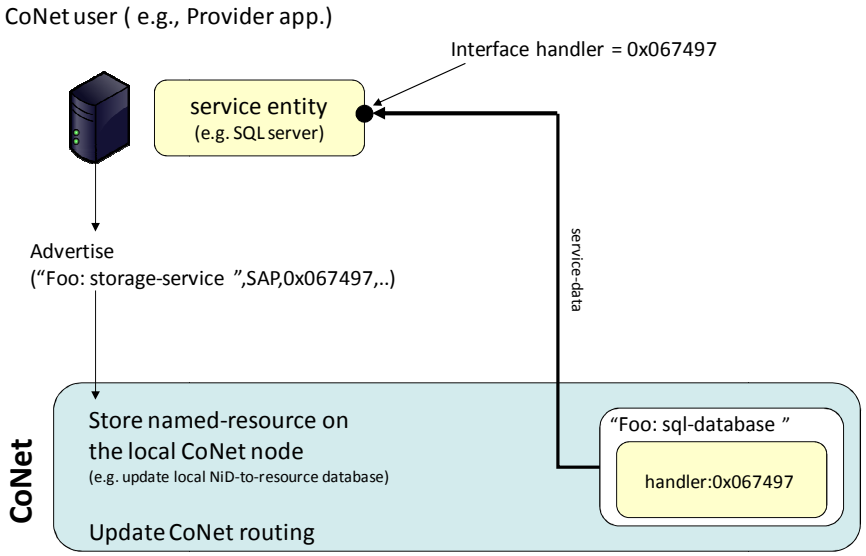


Figure 37: Advertising a local named-resource (named-service-access-point)

7.1.7.3 Downloading data

Figure 38 depicts the case of a CoNet user (e.g. a end-user application) who wishes to download the document identified by the NID “the times: Sept-30-2010”. The user application uses the GET primitive. The local CoNet node identifies the specific instance of the GET primitive with a Location Identifier (LID). Then the CoNet node sends the network data unit containing (at least) the LID and the NID. The CoNet *routes-by-NID* this data unit to the best remote node that is advertising “the times: Sept-30-10”. When the remote node is reached, the CoNet functionality replies directly, sending-back the data-unit, containing the file “times09-10-10.pdf”. This data-unit is routed using the LID and contains the NID (“the times: Sept-30-10”). The presence of the NID in the reply allows intermediate nodes to identify the traversing content and possibly to perform caching.

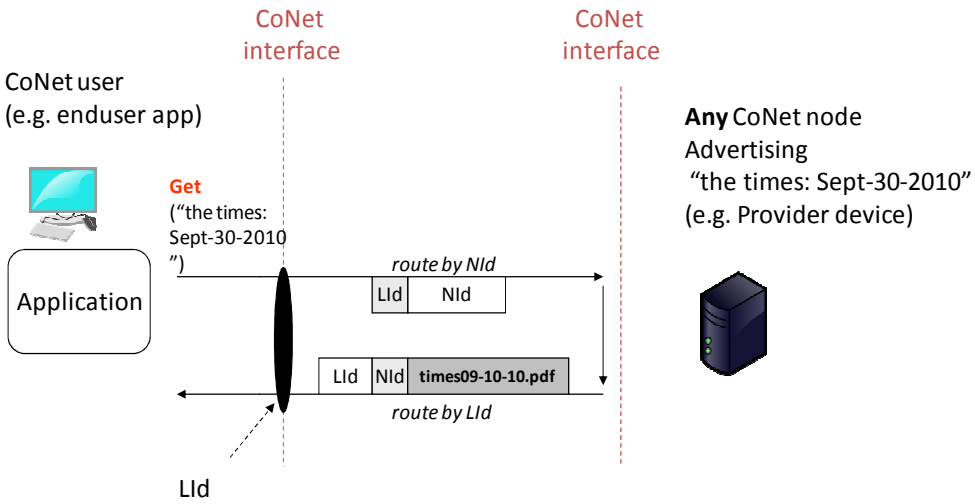


Figure 38: Downloading a named-resource (named-data)

7.1.7.4 Request-response service session

Figure 39 depicts the case of a CoNet user (e.g. an end-user client application), who wishes to send *un-named-data* (e.g. a SQL query)) to a remote server with NID "Foo: sql-database". The user application uses Send2Name, specifying the local entity port 0x0076 where the response is expected. The local CoNet node associates the local entity port with a Location Identifier, LID. Then the CoNet node sends the network data unit-containing the LID and the NID. The CoNet routes-by-NID the data-unit to the best remote node that is advertising "Foo: sql-database". When the remote node is reached, the remote CoNet functionality invokes Indication on the entity listening to the local entity port of the server application, and transfers the LID and the un-named-data, i.e. the SQL query. Subsequently, the server processes the request and replies to the client using Send2Location and including the response (un-named-data), the client LID and the local entity port 0x067497 of the local interface where further replies from the client are expected. The CoNet adds a local-identifier LID* addressing the local entity port and routes by LID the resulting data-units. When the *un-named-data* reach the client, the CoNet functionality of the client device sends an Indication to the client application, which includes both the server response and the LID* of the remote server.

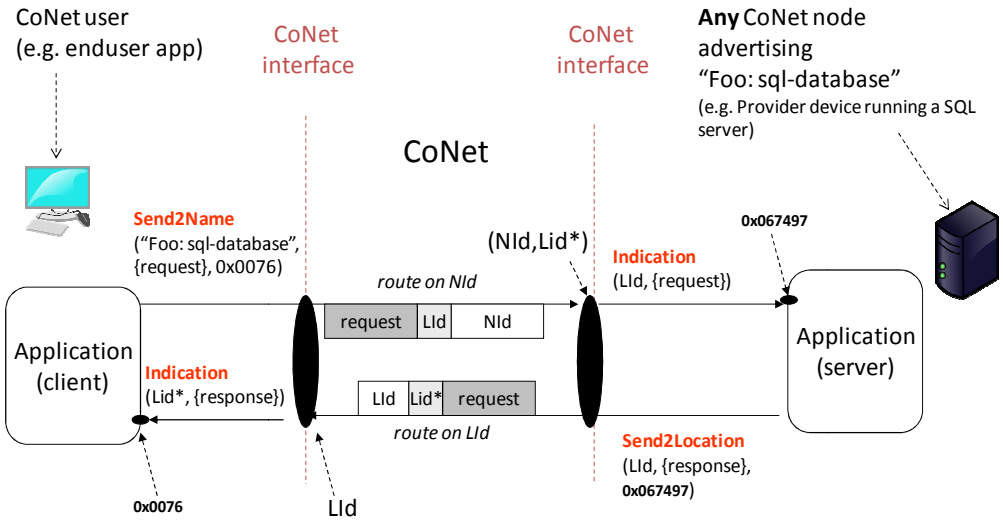


Figure 39: Request-response service session

7.1.7.5 Interactive service session

Figure 40 depicts the case of a CoNet user (e.g. an end-user CoMid entity) who wishes to interactively exchange un-named-data (i.e., middleware messages) with a remote CoMid entity identified by the NID "Foo: mxm-content-provider". This case is equivalent to the case of the request-response service session described in Section 7.1.7.4. However, after the first interaction, the endpoints continue to exchange un-named-data using Send2Location.

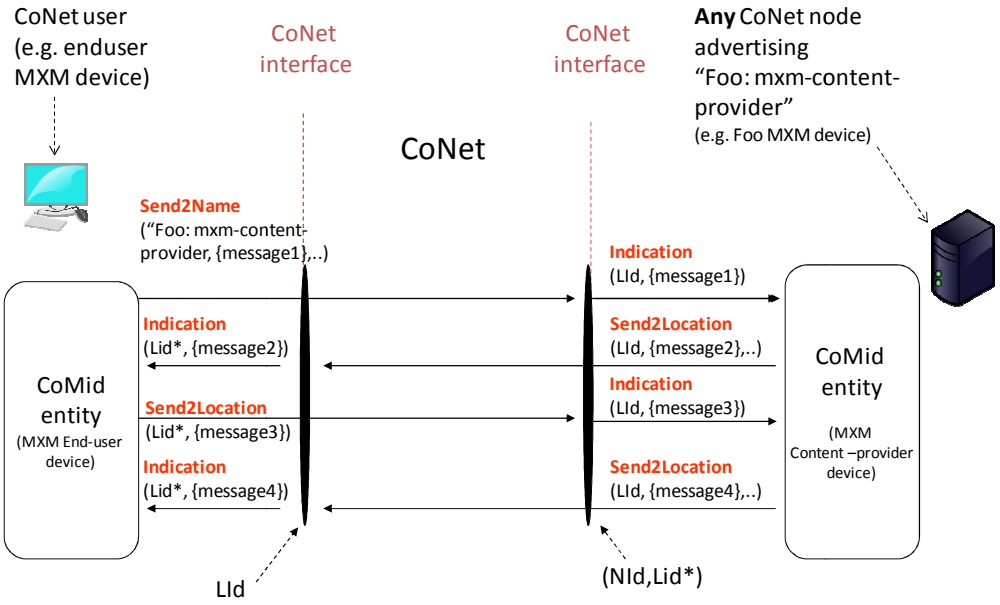


Figure 40: Interactive service-session

7.1.7.6 Remote advertising

The advertising examples presented in sections 7.1.7.1 and 7.1.7.2 describe cases in which the named-resource is provided locally. This means that a user who wishes to provide a named-resource should use a device that is always connected to the network. However, the user may not want to manage such a device and may prefer to use a hosting provider who will take care of advertising her named resources. This can be achieved by using an application service, rather than a CoNet service. Thus, a Foo hosting provider could advertise a service named "Foo: hosting-service" to which the user can "tunnel" her named-resource (i.e., the named-data and the relevant NID). The hosting-service will then advertise (and store) the named-resource locally.

7.2 Security Component (CoSec)

7.2.1 Survey

In the architectural design of CONVERGENCE, CoSec is the component in the Computing Platform level responsible for handling cryptographic protocols and security-related tasks. The CoSec has a *distributed architecture* encompassing several independent and possibly distant components with each component comprising software as well as hardware. In the application flow of security protocols, these components interact with each other. As a consequence, their APIs can be quite complex.

CoSec offers "high-level" security support to CoMid; for instance encryption of content, authentication of users, etc. (Note: it offers similar support to CoNet). In CoMid, the Security TE offers these services to all CoMid engines.

In its current state, the MPEG API for the Security TE is not fully aligned with this architecture. First, the MPEG API offers a variety of low-level cryptographic functionality (such as key-generation, random-number generation), which it would be better to assign to CoSec internal behaviour. Second, CoSec requires significant extensions. This is especially important in scenarios which use smart cards as a secure repository. In this scenario, some security relevant functions would be performed entirely on-card.

We therefore plan to reconsider the API for the current Security TE, define a new API for the CoSec services offered to TEs and specify a second, more complex API for CoSec internal cryptographic functions. The full definition of these API lies beyond the scope of D3.3 and will be completed in subsequent work (see deliverable D5.2).

CoSec components are **distributed on the following computing platforms**:

1. Client computers (e.g. end-user laptops)
2. Application servers
3. Peers (e.g. node computers in a network)
4. Smart Cards (typically held by the end-users)

Typically, execution of *cryptographic protocols* will require interactions between *several* of these. Due to the nature of the protocols and their underlying parameters, the interface *between* CoSec functions will inevitably be complex. A detailed description lies beyond the scope of this document.

7.2.2 Cryptographic Primitives

To implement cryptographic functionalities, CoSec will use *off-the-shelf* solutions:

1. **Fast *symmetric encryption* and *decryption* of content** (F.ENG-VDI c.)
    a. AES-CBC
    b. ABE (Attribute Based Encryption), IBE (Identity Based Encryption) for key derivation. (ABE is still evolving and requires further research work, though)
2. ***Asymmetric cryptography*** (F.USER-AUTH-CONV, F.USER-AUTH-SERV, F.SERV-AUTH, F.PEER-AUTH, F.ENG-AUTH, F.ENG-VDI,F.ENG-VDI-SIGN-THPART)
    a. RSA, for key agreements, signatures, certificates, etc.[15]
3. **Basic primitives (cryptographic hashes etc.)** (F.USER-AUTH-CONV, F.USER-AUTH-SERV, F.SERV-AUTH, F.PEER-AUTH, F.ENG-AUTH, F.ENG-VDI)

[15] (As an alternative, we are also investigating the use of Elliptic Curves

Beyond these standard solutions, some of the Security requirements will require the use of more *advanced* cryptographic primitives. In particular CONVERGENCE will use:

4. ***Group Signature Schemes*** (F.USER-ID b., F-USER-ID-GROUP)
5. ***Identity and Attribute Based Encryption*** (F.TE-LIC-ISS, F.TE-LIC-ENF)
6. ***Pseudonymous access via "Restricted Identification".*** **(F.USER-ID c., F-USER-ID-PSEUDONYM)**

7.2.2.1 Group Signature Schemes

Group Signature Schemes allow any member of a previously created group to digitally sign a document or an object (in the case of CONVERGENCE a VDI or a part of a VDI) on behalf of the entire group. In here, the term group refers to a set of people that perform equivalent tasks and do not necessary need each other's consent (ex. editors of Wikipedia). The group has a single public key that a third party can use to verify if the signature originates from a member of the group. However third parties cannot ascertain the identity of the group member responsible for the signature. If the same member of the group signs on multiple occasions the signatures cannot be linked. In other words, group signature schemes provide not just *pseudonymity* but *anonymity,* except in rare conditions*,* when the group master can legitimately unveil the identity of a user.

Group Signatures are quite a recent development and legal implications/requirements related to their use have yet to be defined. In general, a group is administrated by a trustworthy *group master* who holds a *master key* with which he can register new members, revoke existing members, and unveil the identity of a signer in the event of a dispute. This ability may be a procedural requirement in some applications, where unconditional anonymity is not permitted.

The LMU scenario provides an example of the way a group signature scheme could be applied. In this scenario the group consists of all registered students who have access to at least some of the lectures offered, and all lecturers. Membership requires a *registration* process. During this process, each student receives a smart card that includes both a personal signature key (used to issue personal digital signatures), and a private group signature key that enables the student to generate an *anonymous* group signature, proving that he or she is indeed a valid member of the corresponding group of students. For instance, a student who proves she is part of a group attending a course, can sign annotations to course materials.

7.2.2.2 Identity Based Encryption – Attribute Based Encryption

Recent years have seen significant progress in so-called Identity Based Encryption (IBE) and Attribute Based Encryption (ABE). Cryptographic protocols based on these schemes have now matured to a point in which they can be deployed in real life applications.

IBE provides asymmetric cryptography (encryption and signatures) without the burden of certificate administration typically associated with the creation of a Public Key Infrastructure (PKI). By using IBI CONVERGENCE can offer encryption and decryption of VDIs without establishing a cumbersome PKI.

Attribute Based Encryption (ABE) makes it possible to map complex *licenses* to *attributes*. The basic idea of an Attribute Based Encryption scheme is that content (or more precisely a content key) is encrypted according to determined attributes. In this way, only key holders with the attributes required by the encrypted content will be able to decrypt it. This avoids the risk that fraudulent users could bypass licensing by tampering with security-related hardware and software: while a license check may be bypassed (for instance through manipulation of proprietary hardware like an e-book-reader), recipients of ABE-encrypted content who do not satisfy specified attributes will be unable to decrypt it.

7.2.2.3 Pseudonymous Access – "Restricted Identification"

In the context of this report, we use Pseudonymous Access as a synonym for the specific "*Restricted Identification*" technology used in the German Electronic Passport: a technology that allows registered users to derive pseudo-identities from a personal private key contained in their smart cards, and where necessary, allows administrators to revoke or black list these identifies. The same technology can also be used to derive *"sector"-specific* pseudonyms, where a sector could be a geographical area, a field of application, a set of transactions with a particular provider etc. This means that, in a given sector, an individual user will always use the same pseudonym but that in other sectors she will have different pseudonyms, which cannot be linked.

The CONVERGENCE smart retailing trial will use Pseudonymous Access to introduce data protection and improve privacy for users of the system. In this scenario, customers will use their shop specific pseudonyms to order goods without revealing their true identity. Pseudonymous Access will prevent shops from cross-linking their data and prevent third parties from mining data from different shops to profile individual customers.

7.2.3 Smart Cards in CONVERGENCE

Cryptographic software, or at least the essential components of cryptographic software, needs to run in a trustworthy environment. In theory, we could assume that the entire network is safe, and that registered devices (laptops, PCs, etc.), once checked, will remain secure devices and can be trusted. In reality, however, networks can be anything but secure, and devices (even if initially correct) can be tampered with by fraudulent users, or manipulated by third parties.

Many of these problems can be solved, or at least mitigated, by the use of *smart cards* as *secure hardware modules*. Smart card software is almost completely safe against tampering. This means that an attacker will find it very hard to access confidential content stored or processed on a well-designed system. In this way, the smart card can serve as a reliable outpost for the Service Provider, which continues to control it, even though it is permanently in the hands of an end-user.

Smart cards can also contain secret information, which is not available anywhere else, even on the manufacturer's or card issuer's site. A typical example is a signature key: such keys are generated *on-card*, are *unknown* to *any* outside entity, and always remain in the physical possession of the card holder. The card-holder can therefore be certain that no one - not even the Service Provider or the card manufacturer - can forge her signature without physical access to her smart card.

7.2.4 Cryptographic Protocols for specific Security Requirements

7.2.4.1 Key generation – User Registration

In the CONVERGENCE user scenarios, users have to register. Registration is administrated by an *Identity Provider*. The registration process involves two steps: *identification* and cryptographic *key generation*.

On registration, each end-user receives a *smart card* containing at least the key sets listed below.

A unique *signature key* pair

The key is generated *on the smart card* itself, *during registration*, according to parameters agreed upon by CONVERGENCE. The secret part of the signature key is stored in such a way that it can *never* be read out of the smart card by any entity whatsoever.

The key is *certified* during registration[16]; the certificate (containing the public key part) is signed by the registration authority and stored on the smart card, as well as on a CONVERGENCE server.

The generation of this key pair is *mandatory* for any user wishing to use a smart card as secure repository.

A unique *encryption key* pair

Each user is assigned a private-public key pair, generated according to the same policy chosen for signature key generation. The encryption key pair is different from signature key pair: using the same key pair twice is prohibited; the processes used for generating the two key pairs must be entirely independent.

The key is certified during registration; and the certificate is stored on the member's smart card, as well as on CONVERGENCE servers.

The generation of this key pair is *mandatory* for each user wishing to use a smart card as a secure repository.

[16] This requires a protocol allowing verification of the correctness and possession of the private key part without having access to it. A simple example of such a protocol consists in obtaining a signature over a random challenge of the verifiers choice under supervision.

A unique group signature key

During registration, each member's smart card generates a random seed, securely transmitted to the group administration authority for computation of the complementary secret key part. The user's includes both parts, both of which must be safely stored on his or her smart card.

Note that the group signature key, unlike the signature key and encryption key, is known not just to the smart card but also to the "group master" (see paragraph above.)

A unique private key for Identity/Attribute Based Encryption

Like group signature key discussed in the previous paragraph, the private keys for users of Identity Based Encryption and Attribute Based Encryption requires are derived from a "master" key held by a trusted authority. The setup is similar to the setup for the group signature scheme.

7.2.5 Authentication of users

Authentication is required by F.USER-AUTH-CONV, F.USER-AUTH-SERV, F.SERV-AUTH, F.PEER-AUTH, F.ENG-VDI and can be achieved through:

a. Challenge-Response protocols involving signatures
b. Key-agreement protocols that do not involve end-user signatures

7.2.6 Integrity/Authenticity

Integrity and authenticity are required by F.ENG-VDI, O.VDI-INT, O.VDI-AUTH, F.ENG-INT, F.ENG-AUTH and may be achieved through:

a. Signing
b. Hash-tables (e.g. containing identifiers and hash-values) stored in a trustworthy service

7.2.7 Licensing

Licensing involves the issuing of licenses (F.TE-LIC-ISS) and license enforcement (F.TE-LIC-ENF). These processes involve:

a. The License Technology Engine, using REL
b. On-card license validation, using Card Verifiable Certificates
c. Attribute Based Encryption, mapping license conditions to attributes

7.2.8 Protocols

7.2.8.1 Standard procedure for end-user authentication to an application:

When an end-user wishes to authenticate herself to an application, she executes the following procedure:

1. She presents her user certificate for the current application (obtained from a Service Provider) to the Application Server
2. The Application Server sends its own certificate (issued to him by an Identity Provider) to the client
3. The client validates the Application Server's certificate.
4. The server performs server authentication if required by the client (including certificate validation by the client).
5. The Application Server validates the user certificate:
    a. Validating the issuing Service Provider's signature and checking for attributes encoding access rights, privileges, etc.
    b. Validating (if not already done) the Service Provider's own certificate (with the public key from the Identity Provider's root certificate), obtained from the SP's database, or from the end-user's smart card.
6. The Application Server requests a client authentication based on challenge-response
7. Both parties continue with Secure messaging, if required by one of the participants.

Remark: If an application has previously validated the certificate of a Service Provider and labelled its public key as trustworthy, Step 5.b. may not be required.

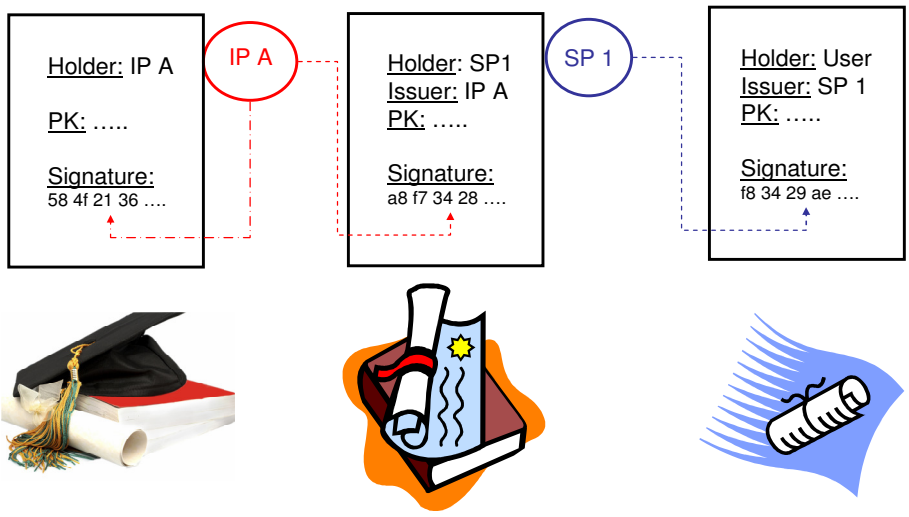
Remark: It may also be admissible – or even required – for an end-user to authenticate under her "real" identity as issued to her by an Identity Provider.

Remark: The public key of Identity Providers is considered to be trustworthy, i.e. we assume that they are contained in root certificates, which inherit their trust through other channels and not through yet another certificate.

Certificate chains:

The diagram below illustrates a simple certificate chain consisting of three certificates:

1. The (self-signed) root-certificate of Identity Provider IP A
2. The Service Provider SP 1's certificate, signed by IP A
3. An end-user's certificate, signed by SP 1



Note that the diagram does not show the "real-name" certificate that the end-user received from IP A and presented to SP1 to obtain the right-hand side pseudonym certificate: validation of this certificate is not part of the chain.

The full *validation chain* is composed of three steps:

1. Validation of the root-certificate (integrity check with attached signature[17])
2. Validation of SP 1's certificate (with public key extracted from IP A's certificate)
3. Validation of User's certificate (with public key extracted from SP 1's certificate)

Provided that a verifier (e.g. a CONVERGENCE application) trusts the IP A to properly execute its duty, these steps will be enough to establish trust in the user's public key and accompanying attributes *without previously having established trust with any Service Provider*.

Remark: Although this chain looks artificially simple, real-world certificate chains are often not much longer. Chains with three hierarchy levels are common.

[17] Since the root certificate is self-signed, this does not establish trust

7.2.8.1.1 Alternative authentication through an Authentication Provider

The following procedure for registration and authentication is adapted from MPEG-M. The procedure uses a third party as Authentication Provider, and is mainly suited for authentication by passwords or verification of a user's biometric features. A drawback of the scheme is that the relation between the Authentication Provider and the Application Provider has to be absolutely trustworthy.

User registration

User registration is a common procedure in almost all modern applications, especially applications that exploit user preferences to select their content. During registration, users provide the personal information needed by the application. This personal information is then linked back to them using an authentication procedure. The result is the assignment of a unique identifier to the user that distinguishes her from other users with different preferences. This identifier may be her e-mail address, a name defined by the user or a name generated by the application.

CONVERGENCE specified a user identification protocol (more details on this protocol will be provided in deliverable D5.2[18]), which takes as input specific data associated with the user to be identified. This data could be a fingerprint scan, an iris scan, a digital certificate, a password, etc. The choice of what data to use depends in some cases on the user, in others on the applications. On success, this protocol returns a user identifier, chosen in line with the policy of the Identity User SP. The user identifier may be a random string, the user's e-mail, etc.

One key issue here is the correctness of the personal data provided as input to the Identify User SP For example, a user might use a false fingerprint to prove her identity. At this point, it should be clear that the Identify User SP is not a registration authority in the Public Key Infrastructure (PKI) sense and that it is still possible to use a PKI or a similar structure to ensure that the user is indeed who she claims to be. Thus, when the user provides her fingerprint, the Identify User SP may require that this fingerprint is signed by a trusted authority (e.g. the user's national police department). In some cases, a trusted third party can provide the Identify User SP with a SAML (Security Assertion Markup Language) assertion, guaranteeing for the user, before the SP issues the identifier. In this way, the third party acts as an authority trusted by both the application and the user, making it possible to "hide" the real identity of the user behind an "anonymous" identifier, while still allowing her to perform critical operations, such as money transactions.

[18] CONVERGENCE has also contributed an input document on user identification protocol to MPEG-M Part 4, which has been accepted and will appear in the next version of this document.

User authentication

The goal of the user authentication procedure is to confirm that the user is who she claims to be. It is thus closely linked to the registration procedure. The authentication scheme is based on SSO (Single sign-on). In this scheme, the task of authentication is assigned to a trusted third party that verifies the user's identity, based on the credentials she provided during registration, and provides her with a token that the application uses to decide whether or not to grant her access, and to define her access privileges.

The Authenticate User protocol, the details of which will be described in deliverable D5.2, uses SAML to perform the user authentication. In CONVERGENCE, an authentication provider uses the Authenticate User elementary service in the middleware to perform the authentication. The authentication (or SSO) provider can thus be seen as an application, used by other applications to perform the authentication in their place. When a user tries to access a restricted area of the application, the application redirects the user to the SSO provider, which performs the authentication. The authentication result is a SAML assertion, passed to the application. The application then decides whether the user will get access to the requested area.

7.2.8.2 Standard procedure for signature issuance and verification:

As in the authentication scenario, each end-user, Service Provider and Application Server has one or more signature key pairs.

Service Providers and *Application servers* only sign with key pairs certified under their *real identity*. Corresponding certificates are issued to them by Identity Providers.

End-users may sign on behalf of their registered pseudonyms. The corresponding certificates are issued by *Service Providers*. They may also sign using their real identity, again requesting the corresponding certificate from a Service Provider. *Verifying* a signature involves certificate validation, and works in the same way as authentication. Only after successful certificate validation can the verifier of a digital signature be sure that a signature has really been issued by the entity claimed. Each verifier of a signature is responsible for establishing this trust.

7.2.8.3 Standard procedure for encryption and decryption:

Since it is impossible to use the private key embedded in the smart card for decryption or for purposes such as signature and authentication, *Encryption* and *decryption* with asymmetric key pairs requires two keys.

Service Providers and *Application Servers* can obtain certificates from Identity Providers.

End-users can obtain certificates for their real names and pseudonyms from *Service Providers*.

An entity wishing to encrypt a message to a public key of another entity has to validate the designated recipient's corresponding certificate *prior* to using its public key for encryption. This means they need to obtain the appropriate certificates from server databases, or obtain them via mail or other means.

Certificate validation involves the same kind of chain validation previously described for the authentication procedure:

Typically, an end-user has to:

1. Validate the recipient's certificate
2. Validate the certificate of the Service Provider who issued the recipient's certificate

Recommendation to thwart "man-in-the-middle-attacks":

It is strongly recommended not to encrypt any message to a public key without having validated its accompanying certificate chain. This is important in particular when encrypting VDIs and uploading them to a cloud server!

Each sender of a message is responsible for establishing trust in the recipient's public key.

7.2.8.4 Key generation of asymmetric key pairs / Certificate storage

Each asymmetric key pair must be accompanied by a certificate, containing the public key of the holder.

Identity Providers:

Identity Providers hold an asymmetric key pair they use to *sign certificates* they issue (certificates issued on behalf of end-users, Service Providers and Application Servers).

An Identity Provider's certificate-signing key pair is considered to be its "master key". Compromising this key would upset the entire system of trust depending on it. The private part of the key is thus the Identity Provider's "sanctuary" and needs to be stored in an absolute secure way. The public part is published in the root certificate of the Identity Provider.

Identity Providers may also have additional key pairs for "ordinary" tasks such as authentication, signing (of VDIs, messages, *not* certificates), and decryption.

Identity Provider	Purpose
"master" key pair	***signing / issuing certificates***
[optional further key pairs]	*authentication/signing/decryption*

Identity Providers will maintain a *database* containing all certificates they have issued, namely:

1. End-users' "real name" certificates
2. Service Providers' certificates
3. Application Providers' certificates

This database is needed for purposes such as revocation, renewal, duplicate enrolment checks, etc.

Service Providers:

Like Identity Providers, *Service Providers* hold an asymmetric key pair for *signing certificates* they issue to end-users.

A Service Provider's certificate-signing key pair is his "master key". Compromising the key will upset the system of trust for all Application Servers and end-users relying on certificates under the Service Provider's authority. The private part of the key is thus the Service Provider's "sanctuary" and needs to be stored in an absolute secure way.

Service Providers may also have additional keys for "ordinary" tasks like authentication, signing (of VDIs, messages, *not* certificates), and decryption.

Service Providers need to be identified under their "real" names.

Service Provider	Purpose
"master" key pair	***signing / issuing certificates***
[optional further key pairs]	*authentication/signing/decryption*

Service Providers will provide a *database* containing all certificates they have issued, namely:

1. End-users' "pseudonym" certificates, along with an entry of the end-users' real identities
2. Application Servers' certificates (optional)

The Service Provider will use the database for purposes such as revocation, renewal, duplicate enrolment checks (in cases where duplicate enrolment is prohibited), and tracing the real identities behind pseudonyms (when requested by an authorized party). Note that the database must not be publicly available, otherwise it would be possible to link pseudonyms to the real identities behind them. *Only Service Providers should be allowed to do so, not even Identity Providers!*

Application Servers:

Application Servers hold asymmetric key pairs in very much the same spirit as end-users.

In CONVERGENCE, it is a requirement that a client connecting to an application should be able to authenticate the server to which she is connecting. CONVERGENCE therefore requires that all Application Servers should hold a certified authentication key pair. The corresponding certificates may be issued by Identity Providers, or optionally by Service Providers.

Note that Application Servers need to be identified under their "real" names.

Application Server	**Purpose**
Authentication key pair	*server authentication*
Signature key pair	*VDI signatures*
Encryption/Decryption key pair	*decryption*

End-users:

End users hold asymmetric key pairs for tasks like authentication, digital signing and encryption/decryption (apart from their key pair(s) originally obtained during registration by an Identity Provider).

The private part of these key pairs is generated and permanently stored on end-users' smart cards. These smart cards are secured in such a way that a private key part cannot be output and will therefore never leave the card.

The end-user's card stores the following key pairs for each Service Provider it uses. In the example below there are three. Note that some fields may be empty (for instance, if encryption/decryption is not required for certain applications).

End-user	"real-name" key pair	SP 1	SP 2	SP 3
Authentication key pair	*authentication towards SPs*	*authentication before Appl. 1*	*authentication before Appl. 2*	*authentication before Appl. 3*
Signature key pair		*VDI signature issuance*	*VDI signature issuance*	*VDI signature issuance*
Encryption/Decryption key pair		*Decryption of received messages/VDIs*	*Decryption of received messages/VDIs*	*Decryption of received messages/VDIs*

Note that there is only one entry for an end-user's "real-name" key pair, and that this key pair is used for authentication purposes only. Its main use is to authenticate the user to Service Providers when requesting certificates from Service Providers' for his (the user's) own key pairs.

7.2.8.5 Key generation procedures / Privacy concerns:

Security Requirement:

All asymmetric key pairs associated with Identity Providers, Service Providers, Applications Servers, and End-Users shall be generated by the entities owning them.

[*Remark:* asymmetric key pairs used to guarantee network security (e.g. for peer authentication) may also be provided by trustworthy providers.]

The corresponding private key parts are kept by their respective owners and *never delivered to any other user.* In particular, end-users' private keys are generated and stored only on their smart cards, and are never an output of any function. This implies that not even the end-user herself will ever know her own private key.

Privacy:

The fact that a party issues a certificate for a key pair does *not* and indeed *must not* imply that the same party issues the key pair. Parties issuing certificates should verify ownership of private keys using cryptographic protocols such as challenge-response or key agreement. Thus Identity Providers, who occupy the highest level in the CONVERGENCE trust hierarchy, do not know any private key except the keys they directly own! This means they are not able to forge signatures on behalf of any entity they certify. This is an important *privacy* issue. Although end-users need certificates issued by Identity- and

Service Providers, they need to be certain that these providers cannot sign messages on their behalf, or decrypt messages encrypted to their public keys, or authenticate under their name or pseudonyms.

Key pair generation:

Key pair generation will follow established standards, and will meet current security requirements with respect to choice of algorithm, key length, parameters, etc. *Smart card issuers* will be responsible for securing cards in such a way that they respect regulatory requirements for key-length, key generation algorithms, etc. The card issuer can also enforce additional conditions for key pair generation. For example, it can bind the generation of specific key pairs to a *role* (to be verified by role authentication) so that only legitimate terminals can activate key generation.

CONVERGENCE assumes that key generation by *Service Providers* and *Application Servers* complies with established policies. In a real-world scenario compliance could involve auditing by certified labs or governmental organizations.

7.2.8.6 Use of keys, owner authorization

End-users hold their private keys on their smart cards, and have no access to these keys. Strictly speaking, this implies that it is not the user who performs authentication signing or decryption, but the smart card.

In order to safeguard against loss or misuse of smart cards, private key operations require authentication of the card-holder, before they are performed.

This authentication is usually performed using

- PIN authentication or
- Biometric authentication with on-card verification

The CONVERGENCE smart cards used in the first phase of the trials will use PIN authentication. A retry-counter will limit false PIN entries to a reasonable number.

Other entities like *Identity Providers, Service Providers, Application Servers,* etc. not using smart cards for their private keys are recommended to store private keys in a *secure repository* (implemented by hardware and/or software) allowing a meaningful authorization (e.g. by passport, hardware token, etc.).

7.2.9 PKI Infrastructure

CONVERGENCE security is constructed on a PKI architecture in which certificates provide a standard means to inherit trust. In certain applications (detailed in future deliverables), PKI will be replaced with IBE.

Very basically, certificates help to perform two basic functions. They

1. *Bind a public key to an entity* (human user, device, network node, institution, etc.) in a trusted way.
2. Allow a trustworthy authority to *associating attributes* with entities.

Certificates are issued by Central Authorities (CAs), whose defining property is that users trust them.

*Being a CA is a **role** that an entity can assume based on specific and agreed policies for a specified group of users.* Thus a CA may be a governmental organization responsible for the citizens of its country; or a manufacturer acting on behalf of a group of retailers and customers, or even an individual user who acts as a CA for a group of co-users (usually a small group).

In CONVERGENCE there will be two kinds of Certificate Authorities (CAs):

- Identity Providers
- Service Providers

CONVERGENCE certificates will comply with the X.509 standard. Each certificate will be associated with:

- A *holder*: the *subject* (= owner) for whom the certificate has been issued
- An *issuer*: the entity that issued (most importantly: signed) the certificate for the owner
- The *holder's public key* (accompanied by attributes referring to the associated algorithms used; e.g. "RSAwithSHA256"
- The *issuer's signature* (accompanied by information referring to the corresponding algorithm used for signing)
- Additional *attributes*, referring to the holder and/or issuer

7.2.9.1 PKI for user registration procedure

User key pair categories:

Each CONVERGENCE user will be associated with three categories of asymmetric key pairs:

1. ***Authentication key pairs***
2. ***Signature key pairs***
3. ***Encryption/Decryption key pairs***

A Key pair belonging to one of these categories must not be used for a purpose defined in another category. For example, an authentication key must not be used to sign a VDI or to encrypt a message to its public key part.

CONVERGENCE smart cards will be programmed to respect these policies, guaranteeing that each key is only invoked for its intended purpose.

The table below illustrates the ways in which these key pairs will be used.

	Application Servers	**Network Peers**	**Client Computers**	**Smart Cards**
Authentication	*SK: server authentication* *PK: user authentication*	*SK + PK: network node* *authentication*	*PK: server authentication*	*SK: user authentication* *[optional:* *PK: Role authentication]*
Signature	*SK: VDI* *signature issuance* *PK: signature/certificate validation*	*VDI* *wrapping signature*	*PK: signature/certificate validation*	*SK: VDI* *signature issuance* *PK: CV-certificate validation*
Encryption/Decryption	*SK: decryption* *PK: encryption*	*decryption/* *encryption*	*PK: encryption*	*SK: decryption*

On the *end-user side*, all private keys can be stored on the client PC. However, it is highly recommended that they should be stored on a smart card.. The smart card is considered as a secure hardware module, which the end-user connects to his client PC as a secure “crypto box”.

Given that network peers will have to support very high throughput communications among network peers, they will not use smart cards, which could limit their performance. In what follows, therefore, we will assume that *server* and *network machines* operate in safe environments.

Below, we will focus on user registration. Networking issues will be treated in D5.2.

7.2.9.2 Registration of users / Roles of Identity Provider, Service Provider and Application Servers

(End-) users will be registered:

1. *Once* by an *Identity Provider*, under their "real identity"
2. By various *Service Providers* for different applications, possibly under different pseudonyms.

Registration of an end-user by an *Identity Provider*:

Registration of End-users, Service Providers, and Applications will be the responsibility of Identity Providers.

When registering to a *CONVERGENCE Identity Provider* that he completely trusts, a user (an end-user, a Service Provider, an Application) will have to reveal her real individual identity (name, address, passport data). The Identity Provider will issue a unique user identifier on behalf of the user, supervise asymmetric key pair generation, issue smart cards, and issue a certificate. Each user may only register *once* ("duplicate enrolment" is forbidden). The Identity Provider will be responsible for verifying a users' claimed identity (e.g. by checking his passport), as well as for revocation of certificates, and duplicate enrolment checks.

A "real world" infrastructure would require complex certificate chains and procedures to verify Identity Providers' public keys (e.g. audited personalization during smart card manufacturing, publication of keys over trustworthy public channels, personal handover, etc.). In the trials, we will assume, for the sake of simplicity, that these keys are certified by self-certifying root certificates.

Registration of an end-user by a *Service Provider*:

CONVERGENCE Service Providers are responsible for assigning specific user identities for use with specific applications. Such identities may be pseudonyms that hide a user's real identity from the application, or real identities.

Service Providers' public keys are certified by a CONVERGENCE Identity Provider; the same holds for the public keys of Application Servers. [Note that we do not allow Service Providers or Application Servers to register under pseudonyms.]

When users register with a Service Provider they do not have to use the same pseudonym they have used with other Service Providers. This avoids cross-linking. Detailed conditions may be subject to policies in place for specific Service Providers and Applications.

End-users can also apply to Service Providers for certificates on key pairs they use for signing or for decryption/encryption. Identity Providers will only issue certificates for authentication key pairs.

Each CONVERGENCE application will cooperate with one or more CONVERGENCE Service Providers it trusts (e.g. with whom it holds contractual agreements). Service Providers may cooperate with more than one application, and applications may cooperate with more than one Service Provider.

7.2.9.3 Certificate Infrastructure

The CONVERGENCE demonstrator architecture will support *three* levels of certificates:

1. (Self-certified) ***Root certificates*** for each CONVERGENCE Identity Provider
2. CONVERGENCE ***Service Provider*** and ***Application certificates*** issued by an Identity Provider
3. ***End-User certificates*** issued by Identity Providers, or Service Providers.

Identity Providers are trustworthy "by definition" (i.e. by assuming an established process ensuring their trust). Their root-certificates are self-signed and contain the public counterpart of the corresponding certificate-signing key.

Service Providers are certified by an Identity Provider[19]. The certificates issued by the Service Provider itself use its own *certificate-signing key* and are used to prove to application servers that the pseudonymous certificates it has issued to end-users do in fact come from a trustworthy Service Provider.

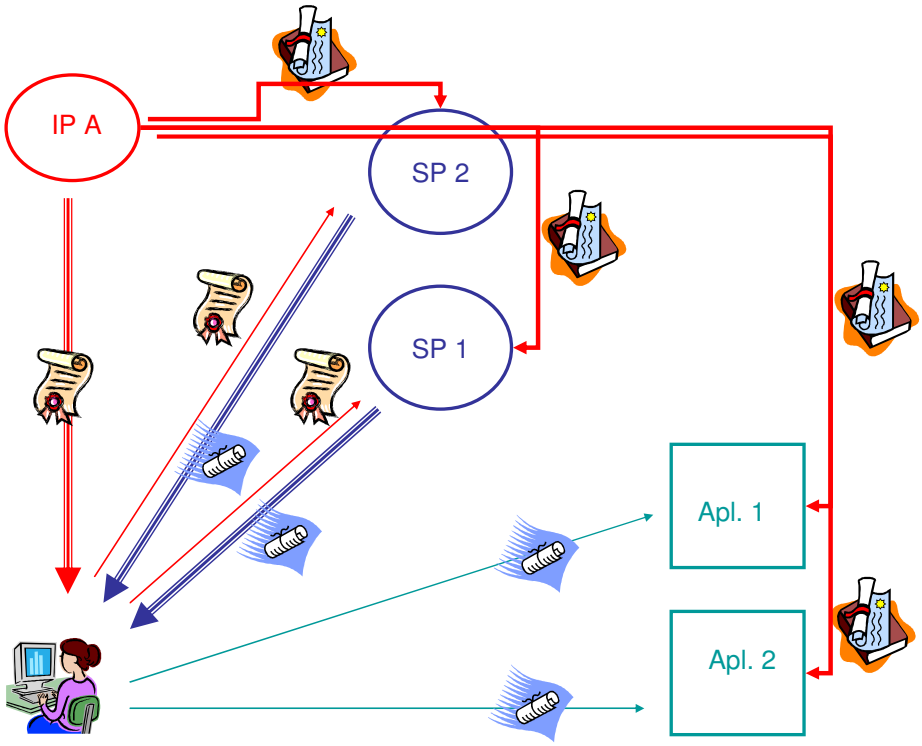
Service Providers may also hold a certificate on its authentication key pair.

Identity Providers also certify the authentication key pairs for *Application Servers* and (optionally) their signing key pairs.

The diagram below depicts the basic infrastructure. In the diagram, *fat* arrows show who issues a given certificate. Thus an end-user obtains her (genuine) certificate from an Identity Provider (IP)[20]. She then presents this certificate (containing her real personal name and data) to two different Service Providers (SPs) that she trusts and requests a pseudonymous certificate from each of them.

[19] In the diagram, for purposes of simplicity, they are all signed by the same provider. In a real life scenario there could be multiple providers.

[20] The corresponding key pair is used for authentication only (mostly for authentication to Service Providers).



In this way, the end-user can apply for certificates on each of her key pairs for each of the three categories described above, i.e. for *authentication*, signature *issuance* and *decryption*.

7.2.9.4 Example

The following table illustrates the content of an end-user's smartcard when she interacts with three different applications.

To begin with, she generates a single key pair that is certified by an Identity Provider – probably with a procedure that requires her to appear in person. The key pair is stored on her smart card and is labelled with the name of the Identity Provider (IP) (see the second column in the table).

Using this certificate she uses online channels to contact various Service Providers, who provide her with certificates for the following additional key pairs she has generated on-card.

1. For Application 1, she requests certificates over key pairs for authentication, signing and encryption from Service Provider 1 (see column three)
2. For Application 2, she needs only key pairs for authentication and signing, and requests certificates on them from Service Provider 2 (see column four)
3. For Application 3, she needs only a key pair for signing, and requests a corresponding certificate from Service Provider 3 (see last column)

She stores the certificates on her smart-card. Optionally, the Service Providers may distribute these certificates to designated databases.

Remark: In this way, end-users' smart cards can serve as decentralized databases for certificates. It is very convenient for an authenticating server to obtain the necessary certificates directly from the requesting end-user, instead of having to store them all or requesting them from a remote database in real time.

Note that the trust in certificates does not stem from the place they are stored or the entities that (physically) dispose of them, but from the *issuer's signature*.

	IP	**SP1** **Appl. 1**	**SP2** **Appl. 2**	**SP3** **Appl. 3**
Authentication	key pair + certificate Key_UIP	key pair + certificate Key_Auth_1	key pair + certificate Key_Auth_2	
Signature		key pair + certificate Key_Sig_1	key pair + certificate Key_Sig_2	key pair + certificate Key_Sig_3
Encryption/Decryption		key pair + certificate Key_Encr_1		

Note that it may be admissible that one and the same Service Provider generates certificates for different applications. In terms of the example below, SP1 = SP2 although Appl.1 differs from Appl. 2.

CV (Card-Verifiable) Certificates:

Some applications may request sensitive data stored on a smartcard (e.g. a stored fingerprint template, medical records), try to store something on the card (e.g. new certificates), or initiate sensitive on-card operations (like key generation). In such cases it is the card itself that ensures that the request comes from a legitimate source, even against the will of the card holder. For instance, a card issuer might encode a health card in a way that only authorized physicians have a right to download medical records. The card will refuse to reveal such records to non-authorized physicians, even if the card holder (the patient) is willing to allow this.

To this end, certificate validation and authentication token validation are performed on-card. This scenario is called "*Role Authentication*". A special (compact!) format for Card-*Verifiable (CV) certificates* supports efficient certificate processing.

As in the off-card scenario, the smart card needs to verify a certificate chain. Root certificates have been stored on-card during personalization by a trustworthy process, and often contain only a root public key rather than a complete certificate. The integrity of such root public keys is ensured by special mechanisms (e.g. writing into secured memory areas, check-sums, etc.).

Validity periods of certificates / Revocation:

In order to avoid a cumbersome structure involving blacklisted certificates, and to ease revocation and renewal, certificates have a limited period of validity. The validity period is part of the certificate, and its proper validation is naturally part of each certificate evaluation.

Each certificate carries at least two dates:

- *Effective date* of generation/issuance
- *Expiration date*

[Side-remark for CV (card-verifiable) certificates: Validating a validity period involves time-checking. Since smart cards do not dispose of a clock, their internal time needs to be updated frequently whenever they contact a trustworthy server. In the case of governmental ID-cards, this is an established procedure].

The normal procedure is to periodically renew all certificates that have not been revoked. The validity periods for existing (expiring) and renewed (replacing the expired) certificates overlap, allowing enough time for distribution.

Typically, within a hierarchical PKI, validity periods are shorter for entities further down in the hierarchy.

8 Bibliography

[1] Network layer solutions for a content-centric Internet. A. Detti, N. Blefari-Melazzi in "Trustworthy Internet", Springer 2010.

[2] A data-oriented (and beyond) network architecture. T. Koponen, M. Chawla, B.G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica in procs. of ACM SIGCOMM'07, 2007.

[3] Networking named content. V. Jacobson, D. K. Smetters, et al., Fifth ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2009.

[4] On the Cache-and-Forward Network Architecture L. Dong, H. Liu, Y. Zhang, S. Paul, D. Raychudhuri in procs. of IEEE International Conference on Communications 2009, ICC 2009.

[5] An Introduction to Petname Systems, Mark Steigler, available at http://www.skyhunter.com/marcs/petnames/IntroPetNames.html.

[6] An IPv4 Option to support Content Networking, A. Detti, S. Salsano, N. Blefari-Melazzi, Internet Draft, draft-detti-conet-ip-option-00, Work in progress, March 2011.

[7] Networking named content, V. Jacobson, et al., in Proc. of ACM CoNEXT 2009.

[8] PURSUIT project website: www.fp7-pursuit.eu.

[9] 4WARD project website: www.4ward-project.eu.

[10] CONVERGENCE Project Deliverable D5.1 "Requirements and Initial Protocol Architecture".

[11] TRIAD: a scalable deployable NAT-based internet architecture, D. Cheriton, M. Gritter, Technical Report (2000)".

[12] Report from the IAB Workshop on Routing and Addressing, D. Meyer, L. Zhang, K. Fall, RFC 4984.

[13] D. Oran, "OSI IS-IS intra-domain routing protocol", IETF RFC 1142.

[14] D. C. Verma "Content Distribution Networks", Wiley-Interscience.

[15] T. Koponen, M. Chawla, B.G. Chun, et al.: "A data-oriented (and beyond) network architecture", ACM SIGCOMM 2007.

[16] D. Smetters, V. Jacobson: "Securing Network Content", PARC technical report, October 2009.

[17] K Katsaros, G. Xylomenos, G. C. Polyzos: "MultiCache: An overlay architecture for information-centric networking", Computer Networks, Elsevier, Volume 55, Issue 4, 10 March 2011, Pages 936-947.

[18] S. Oueslati, J. Roberts, N. Sbihi: "Ideas on Traffic Management in CCN", Information-Centric Networking, Dagstuhl Seminar.

[19] The Many Faces of Publish-Subscribe, P.T. Eugster, P.A. Felber, R. Guerraoui, A. Kermarrec, in ACM Computing Surveys (CSUR), Volume 35, Issue 2, 2003.

[20] W3C HTML 4.01 Specification, Section 12 – Links, 1997.http://www.w3.org/TR/html401/struct/links.html.
[21] ISO/IEC 23006 – Information Technology – Multimedia Service Platform Technologies (MPEG-M).
[22] ISO/IEC 21000-2 – Information technology -- Multimedia framework (MPEG-21) -- Part 2: Digital Item Declaration.
[23] ISO/IEC 21000-3 – Information technology -- Multimedia framework (MPEG-21) -- Part 3: Digital Item Identification.
[24] ISO/IEC 21000-4 – Information technology -- Multimedia framework (MPEG-21) -- Part 4: Intellectual Property Management and Protection Components.
[25] ISO/IEC 21000-5 – Information technology -- Multimedia framework (MPEG-21) -- Part 5: Rights Expression Language.
[26] ISO/IEC 21000-15 – Information technology -- Multimedia framework (MPEG-21) -- Part 15: Event Reporting.
[27] ISO/IEC 21000-19 – Information technology -- Multimedia framework (MPEG-21) -- Part 19: Media Value Chain Ontology.
[28] ISO/IEC 21000-20 – Information technology -- Multimedia framework (MPEG-21) -- Part 20:Contract Expression Language.
[29] ISO/IEC 23006-1 – Information technology -- Multimedia Service Platform Technologies – Part 1 – Architecture.
[30] ISO/IEC 23006-2 – Information technology -- Multimedia Service Platform Technologies – Part 2 – MPEG Extensible Middleware API.
[31] ISO/IEC 23006-4 – Information technology -- Multimedia Service Platform Technologies – Part 4 – Elementary Services.
[32] ISO/IEC 23006-5 – Information technology -- Multimedia Service Platform Technologies – Part 5 – Service Aggregation.
[33] B. B. Mandelbrot, The Fractal Geometry of Nature, W.H. Freeman and Company, New York, 1982.
[34] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, L. Massouli´e, Epidemic information dissemination in distributed systems, IEEE Computer 37 (2004) 60–67.
[35] R. Friedman, D. Gavidia, L. Rodrigues, A. C. Viana, S. Voulgaris, Gossiping on Manets: the beauty and the beast, SIGOPS Operating Systems Review 41 (2007) 67–74.
[36] A. J. Ganesh, A. M. Kermarrec, L. Massoulié, Scamp: Peer-to-peer lightweight membership service for large-scale group communication, in: J. Crowcroft, M. Hofmann (Eds.), Networked Group Communication, volume 2233 of Lecture Notes in Computer Science, Springer Berlin /Heidelberg, 2001, pp. 44–55.

[37] A. M. Kermarrec, L. Massoulié, A. J. Ganesh, Probabilistic reliable dissemination in large-scale systems, IEEE Transactions on Parallel and Distributed Systems 14 (2003) 248–258.

[38] M. Muhr, R. Kern, M. Granitzer, Analysis of structural relationships for hierarchical cluster labeling, in: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10, ACM, New York, NY, USA, 2010, pp. 178–185.

[39] Ahmed, R.; Boutaba, R.; "A Survey of Distributed Search Techniques in Large Scale Distributed Systems," Communications Surveys & Tutorials, IEEE , vol.13, no.2, pp.150-167, Second Quarter 2011.

[40] S. Pantelopoulos, P. Gkonis: "Tools and Sample Applications Plans and Vision" CONVERGENCE public deliverable D7.1, July 2011.

[41] The IMDB Mapping Movie Ontology by Gunnar Grimnes, http://www.csd.abdn.ac.uk/~ggrimnes/dev/imdb/IMDB.rdfs.

[42] National Retail Federation, http://www.nrf.com/.

[43] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. T-Man: Gossip-based fast overlay topology construction. Computer Networks, 53 (13):2321–2339, 2009.

[44] Stoica, Ion et al. (2001). "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications". Proceedings of SIGCOMM'01 (ACM Press New York, NY, USA).

[45] CONVERGENCE Project Deliverable D.7.2.

[46] IPv4 and IPv6 Options to support Information Centric Networking, http://tools.ietf.org/html/draft-detti-conet-ip-option-02.

9 ANNEX A – Survey of solutions of real world descriptors

In terms of real world descriptors, there are now several possible solutions. Two of the most outstanding are NRF ARTS and Etilize.

The Association for Retail Technology Standards (ARTS) of the National Retail Federation is an international membership organization dedicated to reducing the costs of technology through standards. Since 1993, ARTS has been delivering application standards exclusively to the retail industry. ARTS has four standards: The Standard Relational Data Model, UnifiedPOS, XML, and the Standard RFPs (in partnership with NRF). Membership is open to all members of the international technology community, retailers from all industry segments, application developers and hardware companies.

The ARTS Retail Data Model offers two distinct perspectives across the retail business. These are the Enterprise Context, which offers insight into the retail enterprise via three levels within the retail operation (home office, distribution and store levels), and Subject Area Composition, which gives an insight into the retail enterprise via the subject areas which cut across all three levels of retail operation.

The ARTS Retail Data Model currently supports ten of eleven retail business areas. These areas include:

- Merchandise flow management
- Inventory management
- Item and price maintenance
- Point of sale processing
- Tender control
- Store administration
- Customer relationship management
- Sales and productivity reporting
- Ordering (partially supported)
- Workforce Management (partially supported)

Etilize Inc. is a subsidiary of GfK Group founded in 2000 and is the largest product data provider of technology and office supply products in the world with databases on 7 million products in 30 countries and in 20 languages. Etilize provides product data to customers in more countries than any other provider of Information Technology and Consumer Electronics content. It offers product data information, product content and product data services for Resellers, Solution Providers, Comparison

Shopping Engines, Online E-tailers, E-commerce websites, Value-Added-Resellers, Manufacturers and Enterprise customers.

Etilize Inc. publishes standardized, e-commerce and retail store-ready product data for multiple sectors of industry including:

- Information Technology
- Consumer Electronics
- Household Appliances
- Photography
- Home&Garden/ Do-It-Yourself
- Data Capture & Point of Sale
- Office Supplies
- Telecommunications
- Custom SKU Development to fit individual needs.