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

Deliverable Number:	**D5.1**
Contractual Date of Delivery to the CEC:	**31.03.2011**
Actual Date of Delivery to the CEC:	**29.05.2011**
Title of Deliverable:	**Requirements and Initial Protocol Architecture**
Workpackage contributing to the Deliverable:	**WP 5**
Nature of the Deliverable:	**Report**
Editor:	**Angelos-Christos Anadiotis, Charalampos Patrikakis, Iakovos Venieris**
Author(s):	**Angelos-Christos Anadiotis, Leonardo Chiariglione, Angelo Difino, Aziz Mousas, Dimitra Kaklamani, Georgios Lioudakis, Charalampos Patrikakis, Andrea Detti, Giuseppe Tropea, Stefano Salsano, Nicola Blefari Melazzi, Helder Castro, Maria Teresa Andrade**
Abstract:	**This deliverable contains requirements and initial protocol architecture definition for the CONVERGENCE platform.**
Keyword List:	**Requirements, specifications, middleware protocols, network protocols**

Executive Summary

This is the first deliverable of WP5, reporting on the requirements for the CONVERGENCE platform and on the definition of the initial protocol architecture.

The deliverable begins with the requirements identified in deliverable D2.1 and reports the name of the service/operation the system uses to satisfy the requirement, if applicable.

Taking into account the need for definition of protocols to be used for serving the communication between CONVERGENCE entities residing on connected devices and hosts, this deliverable contains the definition of the middleware protocols that will ensure:

1. Support of a standardised interface for the communication among these entities.
2. Provision of services to instances of the CONVERGENCE middleware, regardless of hardware, software, and operating system.

As far as concerns middleware protocols, the WP5 team has decided to base the protocol architecture for the CONVERGENCE middleware (COMID) on the MPEG-M Part 4 [3] protocol suite. The team therefore defined the architecture, specifying it in such a way as to make as much use as possible of the technology and protocol engines defined in the standard, and introducing enhancements and additions when necessary to support specific use cases. After describing the elementary services used to perform basic operations in the MPEG-M architecture, the deliverable goes on to describe the protocols themselves and the associated protocol engines. It then provides a detailed description of the elementary services selected for inclusion in the CONVERGENCE middleware, and describes the protocol for each service.

The section of the deliverable dedicated to network protocols focuses on the specification of the protocols supporting the Convergence NETwork (CONET) and the algorithms that support the CONET API, as identified and defined in [5]. In this initial description of the protocol architecture, the main emphasis is on the protocols and algorithms used to retrieve named-data. Subsequent WP5 deliverables will provide a complete list of protocols including those required for the delivery of un-named data (e.g. the “send-to” primitive in the API), point-to-multipoint data distribution (for video streaming), and data-centric security.

INDEX

1 Introduction

This document presents requirements for the CONVERGENCE middleware and network protocols and specifies the initial protocol architecture. The requirements are based on those previously defined in deliverable D2.1. The deliverable goes on to describe the middleware Protocol Engines. These will be based on an extended version of the MPEG-M part 4 protocol suite. Then, the deliverable presents the network level protocols and the algorithms necessary to support the API for the COnvergence NETwork.

We recommend to read D3.2 “System Architecture”[5], which provides an overview of the CONVERGENCE system architecture, before reading this document.

2 Glossary

Term	Definition
Access Rights	Criteria defining who can access a VDI 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 one way in which the CONVERGENCE System may be used by specific users in a specific situation; more narrowly: a scenario describing the commercial products and services bought and sold in such a situation, the actors concerned and, possibly, the associated flows of revenue.
Clean-slate architecture	The CONVERGENCE implementation of the Network Layer, totally replacing existing IP functionality. See “Integration Architecture” and ””“Overlay Architecture” and “Parallel Architecture”.
CoApp	The CONVERGENCE Application Layer.
CoApp Provider	A user offering Applications running on the Convergence Middleware Layer (CoMid).
CoMid	The CONVERGENCE Middleware Layer.
CoMid Provider	A user providing access to CoMid services. CoMid services may be offered by a single provider or by a federation of providers.
CoMid Resource	A virtual or physical object or service, referenced by a VDI, e.g. a video file, a Real World Object, a person, an Internet service, etc. See “VDI”.A resource can be virtually packaged as a VDI, on its own or together with other resources and metadata. The packaging operation includes the creation of an XML file describing the structure of the VDI and the links among its components.
Community Dictionary	A Functional Block belonging to the CONVERGENCE Middleware Layer that provides all the matching concepts in a user’s subscription,

Service (CDS)	search request and publication. The CDS parses and interprets the ontologies created by the CONVERGENCE users. It thus enables to understand users subscriptions or search requests as well as publications.
CoNet	The CONVERGENCE Network Layer.
CoNet Provider	A user providing access to CoNet services, i.e. it is the corresponding of today's Internet Service Provider.
CoNet Resource	A resource of the CoNet that can be identified by means of a name; it can be either a Named data or a Named service access point.
Content-based resource discovery	A user request for resources, either through a subscription to the CONVERGENCE System or a search request to the CONVERGENCE system. The CONVERGENCE system will then return a list of VDIs compatible with the search criteria. 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). In other terms, the subscription is based on the actual content of the considered events, which are not classified according to some predefined external criterion (e.g., topic name), but according to the properties of the events themselves. See “Subscription” and “Publish-subscribe model”.
Content-centric	A network paradigm in which the network directly provides users with contents, and is aware of which content is actually transported, instead of limiting itself to providing communication channels between hosts.
CONVERGENCE Applications layer (CoApp)	The layer of CONVERGENCE that establishes the interaction with CONVERGENCE users. The Applications Layer interacts with the other CONVERGENCE layers on behalf of the user.
CONVERGENCE Core Ontology (CCO)	A semantic representation of the CoReST taxonomy. 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 bundled together to deliver specific functionality and made available to users and to other Engines via API
CONVERGENCE	A layer of CONVERGENCE that provides the means to handle resources

Middleware layer (CoMid)	on the basis of "what" they contain and offer. These functionalities are implemented using a set of technologies that we call CONVERGENCE Middleware (CoMid) and which includes the Community Dictionary Service (CDS) Functional Block.
CONVERGENCE Network layer (CoNet)	A layer of the CONVERGENCE system that provides access to named-resources on a public or private network infrastructure.
CONVERGENCE Resource Semantic Type (CoReST)	A list of concepts or terms that makes it possible to categorize CONVERGENCE resources, establishing a connection with the resource's semantic metadata.
CONVERGENCE System	A system built by using the technologies specified or adopted by the CONVERGENCE specification. The CONVERGENCE system consists of a set of interconnected CONVERGENCE devices - peers and nodes - connected to each other. See "Node" and "Peer".
Describe, Discover, Distribute (DDD)	A paradigm that uses three different axes to aggregate the pillar functionalities offered by the CoMid layer of CONVERGENCE.
Digital forgetting	Techniques designed to ensure 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 content, content and context related metadata, and structure. The structure is given by a Digital Item Declaration (DID) that links content and metadata.
Discussion	A set or graph of messages each containing links to other messages. In the context of CONVERGENCE discussions and messages within discussions may be represented as VDIs.
Domain ontology	An ontology, dedicated to a specific knowledge domain or application. In CONVERGENCE, domain ontologies will be used to establish associations between VDIs and their semantic metadata. Examples of domain ontologies include the W3C Time Ontology and the GeoNames ontology.
Elementary Service (ES)	A concept imported from the emerging MPEG-M standard. Refers to the most basic unit of functionality offered by the CoMid. ES provide

	CoMid's main functionality. Elementary Services can be used on their own or in combination with CONVERGENCE entities to implement a series of actions.
Entity	An object which an Elementary Service can act upon or with which it can interact. Entities can be VDIs, resources, devices, events, group, licenses/contracts, services and users.
Expiry date	The last date on which a VDI may be legitimately used by a user of the CONVERGENCE System. The last date on which the CONVERGENCE system will allow a user to find the VDI in a search or subscription or to retrieve the VDI.
Fractal	A semantically defined virtual cluster in a distributed overlay network composed of all CONVERGENCE peers running CoMid.
Functional Block	Partial or complete implementations of the functionality required by a specific layer of the CONVERGENCE architecture. Functional blocks are implemented as Engines.
Functional layer	An aggregated set of conceptually similar functional blocks. The CONVERGENCE architecture defines three functional layers: Application, Middleware and Network.
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 itself content-aware. Unlike the clean-state architecture, the Integration Architecture does not imply the need to give up IP. On the other side, it performs better than a CoNet running over IP, as proposed in the overlay approach. See "Clean-state Architecture" and "Overlay Architecture" and "Parallel Architecture".
License	A machine-readable expression of Operations that may be executed by a Principal.
Local named resource	A named-resource made available to the 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 referenced by a VDI, including but not limited to provenance, classification, access rights, 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 standard proposed by MPEG as an extension of the MXM standard.
Multi-homing	In the context of IP networks, the configuration of multiple network interfaces or IP addresses on a single computer.
Named data	A named-resource consisting of data.
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"). Examples of named-resources include VDIs, electronic documents and points of access to a service.
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.
Network Identifier (NID)	An identifier identifying a named resource in the CONVERGENCE Network. If the named resource is a VDI, its NID may or may not be identical to the VDI identifier (to be decided).
Node	A CONVERGENCE device that implements CoNet functionality. Typically, a node is owned by a Telco or Service Provider.
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”
Peer	A CONVERGENCE device that implements CoMid and CoNet functionality. A node will typically be owned by an end-user or by a Service Provider or Telco.
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	From the perspective of the Rights Expression Language, the definition of Principal is: the User to whom Permissions are Granted in a License. Note: Depending on the approach chosen to assign Network Identifiers (NIDs) to CoNet resources, the concept of principal may also be used for such naming purposes. For instance, each principal may be associated with a string (e.g. assigned by a naming authority), and each CoNet resource may have a NID of the form: NID = <namespace ID, hash (Principal), hash (Label)> hash (Principal) 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 that Principal. As an example, the principal can 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.
Publish	The act of making a VDI available to users or to a subset of users of the CONVERGENCE System.
Publisher	A user of CONVERGENCE who advertises resources on the CONVERGENCE system, thus making them available to subscribers.
Publish-subscribe model	A service model based on an asynchronous exchange of messages or events. The CONVERGENCE publish subscribe model, encompasses a set of clients that publish VDIs, which are then forwarded to clients who have expressed interest in receiving them. There are basically two approaches to implementing a publish-subscribe system: 1) topic-based; and 2) content-based. CONVERGENCE adopts the latter. The topic-based approach has 2 variants: list-based and broadcast-based. Both use pre-identified topics to send messages, but

	their implementations differ. In the list-based approach, the system maintains lists of subscribers for each topic and when a message of a certain topic is published, it is sent to the corresponding subscribers. In the broadcast-based approach, when a message is published, the topic of the message is broadcasted to all clients and they decide if they are interested in that message. The Content-based approach does not require previous knowledge of topics. Subscribers indicate their preferences and the system dynamically inspects published content to decide which subscribers should be notified. In other terms, the subscription is based on the actual content of the considered events, which are not classified according to some predefined external criterion (e.g., topic name), but according to the properties of the events themselves [5]. This approach is harder to implement - it requires efficient, scalable algorithms to inspect content and users' interests, and to provide good matches.
Real World Object	An object existing in the real (as opposed to the virtual) world.
Resource	A virtual or physical object or service referenced by a VDI, e.g. media, Real Life 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.).
Service Level Agreement (SLA)	An agreement between a service provider and another user of CONVERGENCE to provide the latter with a service whose quality matches parameters defined in the agreement.
Subscribe	The act whereby a users requests notification every time another user publishes or updates a VDI that satisfies user-defined subscription criteria (key value pairs in the metadata, free text, key words etc.). Note: subscription criteria should be formulated in the same way as search criteria) Subscribe is a CoMid function.
Subscriber	A user of CONVERGENCE who declares his/her interest in being informed about a VDI. A subscriber can declare her interest using keywords, free-text or by indicating specific VDIs.

Subscription	The registration of an interest in a VDI, by a subscriber.
Timestamp	A machine-readable representation of a date and time.
Trials	Organized tests of the CONVERGENCE System in specific business scenarios.
Un-named data	A data resource with no NID.
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). If the service is provided by the application layer the user is an entity outside the CONVERGENCE system, who interacts with the system and uses what the system delivers.
User ontology	An ontology (a set of concepts and their relationships), created by users of CONVERGENCE when publishing a VDI or subscribing to a VDI.
User Profile	A description of the attributes and credentials of a user of the CONVERGENCE System.
VDI Browser	A tool allowing users to browse and consume VDIs on the network as allowed by their access rights.
VDI Creator/Editor	A tool allowing users to create, publish, read, update and delete VDIs.
VDI Identifier	A unique signifier assigned to a VDI or components of a VDI.
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. VDIs can be seen as an extension of the MPEG-21 concept of a Digital Item (DI).

3 Requirements

3.1 Introduction

The requirements listed below were originally defined in Deliverable D2.1. We have modified the last column in the table to include the name of the service/operation that the system uses to satisfy the requirement, when applicable.

3.2 List with Requirements

Area	Sub-area	Code	Requirement	Service/Operation/Notes
VDI capabilities	High level	VDI2	VDIs shall be capable of storing information on the physical location of a resource	Store Content [4.3.19]
		VDI3	VDIs shall offer a range of security/privacy protecting features including but not limited to the definition of access rights, the enforcement of access rights (trials only), automatic authentication (when appropriate), support for encrypted metadata and content, digital signatures, digital forgetting	Authenticate User [4.3.3], Authenticate Content [4.3.1], Create License [4.3.5]
		VDI4	Users shall be able to publish/unpublish a VDI to all users or to a subset of users of the CONVERGENCE System	Create License [4.3.5]
		VDI5	Users shall be able to subscribe/unsubscribe to a VDI or to all VDIs meeting specified search criteria	Revoke Content [4.3.17]
		VDI9	The CONVERGENCE System shall provide for the storage of VDIs on the CONVERGENCE Network	Store Content [4.3.19]
VDI capabilities	VDI metadata	VDI25	A VDI shall be capable of representing attributes of a resource specific to that particular kind of resource	Describe Content [4.3.7]

			(e.g. author, owner, time and date of production, location of production, current location, content information etc.) for a resource	
		VDI26	A VDI shall be able to express attributes of a resource expressed in terms of location (e.g. the location where a photograph was taken). Such attributes shall not be restricted to the current physical location of the resource.	Describe Content [4.3.7]
		VDI27	A VDI should be able to associate a resource with timestamps identifying the beginning, the end or the duration of an attribute of the resource	Describe Content [4.3.7]
		VDI28	A VDI shall be able to associate a resource with typed links to other VDIs (e.g. <derived from>, <name of VDI>; <full version available in>, <name of VDI, <has also been bought by>, <set of VDIs>	Describe Content [4.3.7]
		VDI29	A VDI shall be able to reference a RWO via an RFID tag, a bar code or any other form of unique identifier for the RWO.	Describe Content [4.3.7]
		VDI30	A VDI shall be able to support user annotations in the form of key words, texts, visual icons, spoken text, references to geographical locations, temporal data, etc.	Describe Content [4.3.7]
		VDI31	A VDI shall be able to specify the way user annotations appear in the VDI browser	Present Content [4.3.12]

			(e.g. the visual presentation of annotations to a photo)	
		VDI32	A VDI shall be able to support user-defined tags	Describe Content [4.3.7]
		VDI33	VDIs shall support both public comments (legible to all subscribers to the VDI) and private comments (legible only to one user or a user-defined subset of users)	Create License[4.3.5]
	Creation/ Publishing of VDIs	VDI34	The CONVERGENCE System shall provide users with an easy to use tool (VDI Creator/Editor) allowing them to create, update, publish and unpublish a VDI, beginning with a set of pre-existing resources and/or VDIs and including all necessary metadata, definitions of access rights, tags etc.	Create Content [4.3.40], Process Content [4.3.13], Create License [4.3.5], Process License [4.3.14], Describe Content [4.3.7], Package Content [4.3.10], Store Content [4.3.19]
		VDI35	The VDI Creator/Editor shall allow a user of the CONVERGENCE System to publish/unpublish a VDI to all users or to a subset of users of the system	Create License [4.3.5]
		VDI40	The VDI Creator/Editor shall allow users with appropriate rights to update the metadata and/or the resources of a VDI *after* it has been published. When a user updates a VDI all copies of the VDI accessible to the CONVERGENCE system shall be automatically updated. This includes all copies of the VDI on the CONVERGENCE network and all copies under the support of a CONVERGENCE application that supports local updating of VDIs	Process Content [4.3.13], Process License [4.3.14]

		VDI41	The VDI Creator/Editor shall allow users to change access rights to a VDI *after* it has been published	Process License [4.3.14]
		VDI42	The VDI Creator/Editor shall allow users to send a message (e.g. a warning) to all subscribers to a given VDI	Request Event [4.3.16], Store Event [4.3.20]
	Browsing, subscription	VDI43	The CONVERGENCE System shall include an easy to use tool (VDI Browser) allowing users to browse, acquire rights and subscribe to VDIs, to cancel their subscriptions and to define attributes of their subscription (e.g. willingness to accept messages from a VDI Creator/Editor, expiry date)	Create License [4.3.5], Request Event [4.3.16]
		VDI48	All applications shall allow users with the appropriate rights to download a VDI to a local device	Deliver Content [4.3.6], Request Content [4.3.15]
		VDI49	The VDI Browser shall allow users with the appropriate rights to access the resources and metadata of a VDI	Request Content [4.3.15]
	Search	VDI51	The VDI browser shall allow users to search for VDIs meeting specific criteria (e.g. presence of a specific value for a specific item of metadata) and specify actions to be taken when such a VDI is found	Search Content [4.3.18]
		VDI52	Searches shall be semantic (e.g. a search using synonyms for keywords as well as keywords themselves)	Search Content [4.3.18]
		VDI54	Searches shall support context sensitivity including sensitivity to location (e.g. it shall be possible to search locations stored in metadata by	Search Content [4.3.18]

			proximity to the center of a city or a place of interest or an address)	
		VDI55	It shall be possible to search for a VDI with a given RFID tag or for other comparable identifiers.	Search Content [4.3.18]
		VDI56	It shall be possible for a user to subscribe to a known VDI or to all VDI meeting given search criteria. Users shall be able to request updates related to their subscriptions (e.g. when a subscribed VDI is updated, when a VDI meeting the search criteria becomes available)	Request Event [4.3.16]
	Device/ communications support	VDI60	It shall be possible to transfer a VDI from one device to another as a file or as a stream	Deliver Content [4.3.6]
	Security	VDI62	A VDI shall allow users to define access rights for VDIs	Create License [4.3.5]
		VDI63	A user shall be able to sign a VDI	Create License [4.3.5]
		VDI64	Users shall be able to automatically authenticate a signed VDI before/after they download the VDI	Authenticate Content [4.3.1]
		VDI65	It shall be possible to encrypt specified fields of a VDI including content, comments and metadata	Create License [4.3.5], Process Content [4.3.13]
		VDI66	It shall be possible to create a VDI in such a way that media contained in the VDI is marked with an un-erasable watermark	Create License [4.3.5], Process Content [4.3.13]
		VDI67	It shall be possible to place a digital signature on certain fields of a VDI including resources, comments and	Create License [4.3.5], Process Content [4.3.13]

			metadata	
		VDI68	The VDI creator and the VDI browser shall support the use of "Personal Secure Tokens" for use in authentication. It shall be possible to use the token on any device attached to the CONVERGENCE system.	Authenticate User [4.3.3]
	Privacy	VDI70	Users shall be able to send, share and receive VDIs without revealing their identity	Create License [4.3.5]
		VDI71	The VDI Browser shall allow users to define minimum levels of authentication for VDIs they are willing to access (e.g. refusing to accept anonymous VDIs)	Create License [4.3.5]
	Digital forgetting	VDI72	A VDI shall allow an application that creates a VDI to define an expiry date for legal access to the VDI	Create Content [4.3.4], Create License [4.3.5]
		VDI74	The VDI creator shall allow users to unpublish all copies of VDIs they have published	Revoke Content [4.3.17]
		VDI75	The CONVERGENCE System shall include a standard tool providing automated garbage collection of VDIs residing on the network that have passed their expiry date. This implies that users will no longer be able to retrieve such VDIs.	Revoke Content [4.3.17]
		VDI76	The CONVERGENCE System shall make it possible to design an application that automatically deletes or encrypts information on a VDI whenever a given condition is met (e.g. customer has left store)	Process Content [4.3.13]
		VDI77	The CONVERGENCE system	Request Event [4.3.16],

			shall make it possible to notify other predefined users, each time the VDI is accessed by a user. The notification shall not contain any personal information concerning the user accessing the VDI	Store Event [4.3.20]
Framework capabilities		FRA2	The framework shall provide functionality to generate unique VDI identifiers	Identify Content [4.3.8]
		FRA8	The framework shall have functionality allowing an application (with a named user) to search for a VDI meeting a given set of search criteria (operating on VDI metadata), without referring to a given physical location where the VDI is stored	Search Content [4.3.18]
		FRA9	The framework shall allow an application with appropriate rights to modify/delete a VDI which has already been published	Revoke Content [4.3.17]
		FRA10	It shall be possible for users to define a set of VDIs meeting certain search criteria (e.g. distance between a physical location referenced in the metadata and a second, user-defined location)	Search Content [4.3.18]
		FRA12	The CONVERGENCE System shall support the use of persistent names for VDIs whose referenced resource changes (e.g. a VDI referring to the front page of a newspaper)	Identify Content [4.3.8], Process Content [4.3.13]
	CDS	CDS2	Users of the CDS should be able to select terms from a list of ontologies	Describe Content [4.3.7]
Network capabilities		NET1	The CONVERGENCE Network (CONET) shall	CONET protocol suite [Sec 5]

			distribute and provide access to resources identified in the network by a network identifier (NID); examples of such resources include: a VDI; an electronic document, an image, a source of information with a consistent purpose, the point of access to a service, and a collection of other resources.	
		NET2	It shall be possible to retrieve a resource without reference to the location where it is stored except insofar as such references are required to support the scope and policy routing functionality	CONET routing is based on names rather than on location [Sec. 5.8.1]
		NET3	It shall be possible to replicate and cache a resource with a given NID over different network nodes, while maintaining the same NID on all copies; the CONET shall provide the functionality to access the "best" (e.g. the closest) resource in this set. In other words the NID shall be treated as an anycast address and the CONET shall provide anycast routing functionality	The same named-resource can be cached on and retrieved from different CONET Border nodes, Internal nodes and serving nodes [Sec. 5.8]
		NET4	A Network identifier (NID) may be either a flat name or may have a hierarchical structure (for example to better support the aggregation of names).	The NID format supports both the VLL (flat) and the PLHB (hierarchical) namespaces [Sec. 5.4]
		NET5	CONET routing shall not necessarily rely on a structured naming scheme	The VLL namespace allows the use of a flat naming scheme [Sec. 5.4]
		NET5	Network Identifiers (NID) shall be designed independently of VDI	The NID format is independent of the VDI identifier. However the

			identifiers. COMMENT: this does not preclude using the VDI identifier as a network identifier, if possible and desirable.	NID format is compatible with the format of the VDI identifier. This makes it possible to use the VDI identifier as a NID [Sec. 5.4]
		NET6	The CONET needs to rely on a mechanism to assign unique network identifiers to a resource.	This mechanism is left for future work.
		NET7	The CONET shall support multi-homing, policy routing, and scope (advertising and/or storage in a restricted geographical/admin area) COMMENT: future discussion will examine how far the concept of multi-homing currently employed in IP networks can be adapted for use in the CONET COMMENT: To be discussed; we need to investigate if support for these functions should be provided natively by CONET or if should use COMID, at least for some classes of contents, (e.g. protected content)	Policy routing could be implemented in the Name System. Name System can provide different routing information depending on the requesting node. Multi-homing is natively supported by routing-by-name mechanism since in case a serving-node has different access links, all these links can be used to reach the same NID. A Name System supporting a CONET layer could have a local scope [Sec. 5.8.1]
		NET8	The CONET shall natively support user mobility allowing users to store and retrieve resources independently of their location.	The source-routing mechanism of CONET layer makes it possible to transfer a named-resource to the user independently of the user's temporary location [Sec. 5.8]
		NET9	The CONET shall provide means to support a point-to-point communication between the upper layer entity that is sending un-named resources (service-data) and the receiving upper layer entity	The source-routing mechanism of CONET layer makes it possible to exchange un-named resources between two specific network endpoints. [Sec. 5.8]

			(e.g. a server). In this way the CONET can support services requiring the interactive exchange of service-data. COMMENT: This is required to make it possible to select a node belonging to the anycast group identified by a given NID and to address terminals not identified by a NID. A possible way to do this is to introduce Location Identifier (LID) and use it to identify both terminals and service access points. The LID would be a unicast address.	
		NET10	The CONET shall provide a proper security model making it possible to decide whether a CONET user is allowed to store, retrieve and delete/update a given resource.	CONET layer security solutions will be specified in a subsequent deliverable
		NET11	CONET shall allow the use of a NID to identify the point of access to a service	The routing-by-name mechanism of CONET layer makes it possible to route un-named data toward a service access point identified by a NID [Sec. 5.8]. The related protocol support will be included in a subsequent deliverable
		NET12	The CONET shall offer functionality to advertise to the network that a resource identified by a NID is stored in a specified node or in a specified set of nodes.	The Name system protocol suite will be defined in a subsequent deliverable
		NET13	The CONET shall offer functionality to send a resource to a specified node.	The source-routing mechanism of CONET layer makes it possible to send a resource to a

				specific network node [Sec. 5.8]
		NET14	The CONET shall offer functionality to retrieve a resource identified by a NID	The CONET layer and transport algorithm make it possible to “download” a named-resource[Sec. 5.8.4]
		NET15	The CONET shall allow a user to request a sequence of updates of a resource identified by the same, known NID. A user should thus be able to request a sequence of logically resources such as a. a real time movie, or a sequence of updates to a document) COMMENT: Future work will investigate whether it is desirable or possible for a user to request to retrieve a resource that is not yet available in the network.	Not supported by CONET anymore, but supported by COMID for VDIs
		NET16	The CONET shall offer functionality to revoke the advertising of a resource identified by a given NID, and to delete the resource from a specified storage node	The Name system protocol suite and routing functions to be defined in a subsequent deliverable will specify how to interact with the name system to insert, update and revoke a routing entry that can be used by CONET routing-by-name (lookup and cache) . Inserting a new routing entry in the name system advertises the presence of a named-resource; deleting the entry makes it impossible to find the named-resource, i.e. revokes the presence of a named-resources. The deletion of a named-

				resource from a storage node is a local functionality, which does not involve a specific protocol mechanism.
		NET17	The CONET shall offer functionality to update a previously advertised resource, identified by a given NID and stored in a specific node.	To be supported by the Name system protocol suite to be defined in a subsequent deliverable
		NET18	The CONET shall make it possible to control where (or in which administrative domain) a resource is stored. COMMENT: It is not acceptable that storage could be completely random. This may imply that DHT cannot be used for this purpose.	To be supported by the Name system protocol suite to be defined in a subsequent deliverable. The protocol suite will make it possible to advertise "where" a named-resource is stored
		NET19	The CONET shall support mechanisms to delete a resource, supporting digital forgetting and garbage collection operations. COMMENT: Future work will investigate whether this functionality should be provided natively by CONET or whether it will be performed in cooperation with COMID (for some classes of contents, e.g. protected content)	To be supported by the Name system protocol suite to be defined in a subsequent deliverable. The protocol suite will make it possible to delete a routing entry making it impossible to find a named-resource on the related serving node. Moreover, CONET data units [Sec. 5.6.3.1.2] include expiry time and cache obsolescence information enabling digital forgetting/garbage collection and soft-state removal of a named-resource from cache.
		NET20	The CONET shall support mechanisms to update resources stored in any node. COMMENT: Future work will	Not supported as an "atomic" operation. However, the owner of a named-resource can sequentially update the

			investigate whether this functionality should be provided natively by CONET or whether it will be performed in cooperation with COMID (for some classes of contents, e.g. protected content)	named-resources stored in any serving node. If the obsolete version of a named-resource is present in network caches it will be deleted after its obsolescence time.
		NET21	The CONET shall support the possibility of accessing a source of information through a persistent name, even if the source consists of a changing set of VDIs (name persistence)	A source of information (e.g., a newspaper server) can send out updates of a named-resource (e.g. new releases of the newspaper) and can name each update with the same NID. Therefore, a user may retrieve the last updated named-resource by using the same NID.
		NET22	The CONET does not provide support to access the latest version of a series of related VDIs; e.g. referenced by the same source of information, and to earlier versions of a series of related VDIs. This functionality will be provided by COMID.	
		NET23	The CONET shall provide native support for caching. Unlike a Content Delivery Network, CONET will, in principle, support caching on every node and also on user terminals. Users can retrieve a resource from any node where it is stored. This shall not necessarily require a connection to CONET but only to a single node storing a copy of the desired resource. COMMENT: This requirement is fundamental to support	CONET border node and internal node support caching [Sec. 5.8.3].

			recently proposed schemes such as wireless caching, distributed storage, and recommendation strategies. By making the network aware of the contents that is handling it is possible to shift traffic away from congested regions and balance loads, for instance by offloading cellular networks, or by implementing other distribution strategies beneficial to network operators. This functionality will strengthen network operators' control over traffic generated by so-called over-the-top players (such as Facebook, YouTube, etc),	
		NET24	A network node needs information about the rights to replicate/cache resources.	Information contained in the named-data CIU, i.e. the Cache Obsolescence Period, supports caching operation/policies (if obsolescence = 0, nodes do not have the right to cache). More sophisticated policies could require Middleware support. This aspect will be further addressed in future deliverables.

4 Middleware Protocols Specification

4.1 Scope of the specification

This section contains a first specification of the CONVERGENCE middleware protocols. It should be noted that since the MPEG-M part 4 protocol suite has been evaluated by the project partners to be well suited to CONVERGENCE needs, the specification of the middleware protocols has been performed in a way that builds upon the existing MPEG-M part 4 protocol suite. However, since the CONVERGENCE framework needs extra functionality, we have attempted to define this functionality in such a way that it can fit in the MPEG-M part 4 architecture. This means that the middleware protocols defined by CONVERGENCE can be offered as extensions to the existing MPEG-M part 4 protocol suite. Possible protocols, candidates for standardization, will be chosen at a later stage in the project.

4.2 Middleware Overview

The CONERGENCE middleware is based on the MPEG-M architecture and protocols [1][2][3] as they are evolving to become standards. The protocols are derived from MPEG-M Part 4 [3] which defines a set of elementary services performing basic operations.

Figure 1 shows an abstract architectural scheme of CONVERGENCE. In this scheme, the middleware consists of Protocol, Technology Engines, and of an Orchestrator (the Technology Engines and the Orchestrator are illustrated in Deliverable D3.2). A Protocol Engine is responsible for processing the protocol of the corresponding Elementary Service, each time it is called. A Technology Engine is a component of the CONVERGENCE Middleware providing technological functionalities via a standard API. An Orchestrator Engine is a component of the CONVERGENCE Middleware capable of organizing chains of Technology Engines to provide aggregated functionalities.

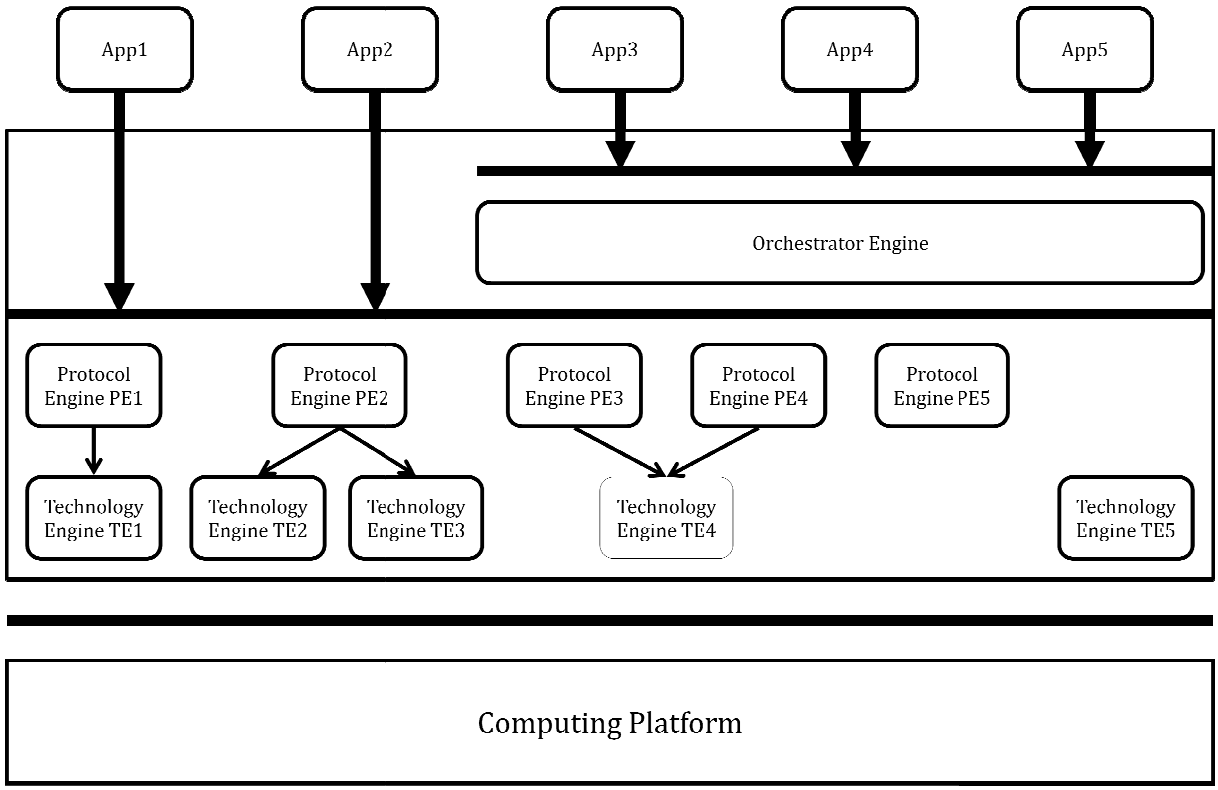


Figure 1: CONVERGENCE Middleware Abstract Architecture

As shown in Figure 1, the Orchestrator is a special engine that lies above the protocol engines. Its role is to act as a service broker for the middleware. Applications can call the orchestrator to setup the chain of engines required to process incoming requests. The orchestrator is aware of all other available engines and of how to access them. At the same time, it provides an abstraction layer to applications and makes middleware operations transparent.

4.3 Elementary Services

4.3.1 Authenticate Content

4.3.1.1 Description

The Authenticate Content elementary service 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 itself, it has to be provided separately.

4.3.1.2 Protocol Specification

This elementary service follows the protocol specification of the MPEG-M Authenticate Content elementary service, defined in MPEG-M Part 4 [3].

4.3.2 Authenticate Device

4.3.2.1 Description

This elementary service enables a user or any client application to authenticate the device where it is running. This way, the owner of the content to be used by the application can be sure that the device is trusted.

4.3.2.2 Protocol Specification

This elementary service has the following protocol specification:

Steps	Client	Service Provider
1.	The client generates an `AuthenticateDeviceRequest` message containing its own certificate or public key and sends the message to the Service Provider.	
2.		The SP verifies the digital signature of the message and responds with an Authentication Success/Failure message.

4.3.2.3 Syntax of Protocol Data Format

```
<!-- ################################################ -->
<!--  Authenticate Device                             -->
<!-- ################################################ -->

<!-- Definition of AuthenticateDeviceRequestType -->
<complexType name="AuthenticateDeviceRequestType">
<complexContent>
<extension base="mpegmb:ProtocolRequestType">
<sequence>
<element name="DeviceKey" type="dsig:KeyInfoType"/>
</sequence>
</extension>
</complexContent>
</complexType>


<!-- Definition of AuthenticateDeviceResponseType -->
<complexType name="AuthenticateDeviceResponseType">
<complexContent>
<extension base="mpegmb:ProtocolResponseType">
<choice>
<element name="AuthenticateDeviceSuccess"
type="mpegm:ProtocolSuccessType"/>
<element name="AuthenticateDeviceFailure"
type="mpegmb:ProtocolFailureType"/>
</choice>
```

```
</extension>
</complexContent>
</complexType>
```

4.3.2.4 Semantics of Protocol Data Format

Semantics of the `AuthenticateDeviceRequestType`:

Name	*Definition*
`AuthenticateDeviceRequestType`	Top-level type for Authenticate Entity request messages. `AuthenticateEntityRequestType` extends `ProtocolRequestType`.
`DeviceKey`	The public key or the certificate of the device to be authenticated, digitally signed.

Semantics of the `AuthenticateDeviceResponseType`:

Name	*Definition*
`AuthenticateDeviceSuccess`	Response in case of success.
`AuthenticateDeviceFailure`	Response in case of failure.

4.3.3 Authenticate User

4.3.3.1 Description

The Authenticate User elementary service allows the client to confirm a user's identity. This service can work with users belonging to different domains using the Single-Sign-On (SSO) mechanism.

4.3.3.2 Protocol Specification

This elementary service follows the protocol specification of the MPEG-M Authenticate User elementary service, defined in MPEG-M Part 4 [3].

4.3.4 Create Content

4.3.4.1 Description

The Create Content Elementary Service enables the creation of a Versatile Digital Item, by providing the data it contains, such as resources, descriptions, licenses etc.

4.3.4.2 Protocol Specification

This elementary service follows the protocol specification of the MPEG-M Create Content Elementary Service, defined in MPEG-M Part 4[3].

4.3.5 Create License

4.3.5.1 Description

The Create License Elementary Service enables the creating of a license. Licenses in CONVERGENCE are expressed using the MPEG-21 Rights Expression Language (REL) [4] along with any possible extensions to be defined in D4.2.

A license may either be created directly, e.g. when it is the owner of a VDI who wants to create it, or it may require additional steps, as when it is created on demand after a transaction. For example, if a user has paid to obtain a license, the service provider has to confirm that the transaction has been successful and then create the requested license.

4.3.5.2 Protocol Specification

This elementary service follows the protocol specification of the MPEG-M Create License Elementary Service, defined in MPEM-M Part 4 [3].

4.3.6 Deliver Content

4.3.6.1 Description

The Deliver Content elementary service enables the client to request the delivery of a specified VDI according to certain 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 these two parties.

Upon completion of the delivery, the service provider may or may not notify the user who requested the delivery that it has been fulfilled.

4.3.6.2 Protocol Specification

This elementary service follows the protocol specification of the MPEG-M Deliver Content elementary service, defined in MPEG-M Part 4 [3].

4.3.7 Describe Content

4.3.7.1 Description

The Describe Content elementary service enables a user to generate, provide and retrieve descriptions of VDI content. This service provides a protocol for setting the description of a VDI, a protocol for getting the description of a VDI and a protocol that supports the generation of rich semantic descriptions of a VDI and its contents. The aforementioned descriptions are also used by the Search Content elementary service in order to perform search functions.

4.3.7.2 Protocols Specification

This elementary service extends the protocol specifications of the MPEG-M Describe Content Elementary Service, defined in MPEG-M Part 4 [3], by adding the Request Named Entity protocol, which is specified below.

4.3.7.2.1 Request Named Entity

This sub-clause specifies the interfaces, protocol specifications, syntax and semantics of the protocol data formats for the Request Named Entity Protocol.

4.3.7.2.1.1 Introduction

The Request Named Entity Protocol is used to request named entities of ontologies (Classes, Instances, Properties) satisfying certain criteria. With the support of the Request Named Entity Protocol, a User can form:

- semantic descriptions, which may be sent to a Describe Content Service Provider
- semantic queries, to search for content using a Search Content Service Provider (SP).

4.3.7.2.1.2 Protocol Specification

The Request Named Entity protocol is as follows :

Steps	Client	Service Provider
1.	A User sends a `RequestNamedEntityRequest` message. This message contains at least a `NamedEntityLabel` and possibly a namespace and entity type.	

2.		If the query specifies a namespace but the Describe Content SP doesn't serve it, then the SP returns a `RequestNamedEntityResponse` with a `RequestNamedEntityFailure` element and the protocol ends here. Else step 3 is performed.
3.		The SP checks for named entities matching user criteria in its repository and returns a `RequestNamedEntityResponse`

4.3.7.2.1.3 Syntax of Protocol Data Format

```
<!-- ################################################ -->
<!--  Request Named Entity-->
<!-- ################################################ -->

<!-- Definition of RequestNamedEntityRequest -->
<element name="RequestNamedEntityRequest"
type="RequestNamedEntityRequestType">
<complexType name=”RequestNamedEntityRequestType">
<complexContent>
<extension base="mpegm:ProtocolRequestType">
<sequence>
      <element name="InputQuery" type="NamedEntityRequestInputQueryType"
minOccurs="1"/>
</sequence>
</extension>
</complexContent>
</complexType>
</element>

<!-- Definition of RequestNamedEntityResponse-->
<element name="RequestNamedEntityResponse"
type="RequestNamedEntityResponseType">
<complexType name="RequestNamedEntityResponseType">
<complexContent>
<extension base="mpegm:ProtocolResponseType">
      <sequence>
<choice>
      <element name="RequestNamedEntitySuccess"
type="RequestNamedEntitySuccessType"/>
      <element name="RequestNamedEntityFailure"
type="mpegm:ProtocolFailureType"/>
</choice>
</sequence>
</extension>
</complexContent>
</complexType>
</element>
```

```
<!-- Definition of NamedEntityRequestInputQueryType-->
<complexType name="NamedEntityRequestInputQueryType">
<complexContent>
<extension base="mpqf:QueryType">
<sequence>
<element name="NamedEntityLabel" type="string" minOccurs="1"/>
      <element name="TargetNamespace" type="anyURI" minOccurs="0"/>
      <element name="EntityType" type="anyURI" minOccurs="0"/>
</sequence>
</extension>
</complexContent>
</complexType>

<!-- Definition of RequestNamedEntitySuccessType-->
<complexType name="RequestNamedEntitySuccessType ">
<complexContent>
<extension base="mpegm:ProtocolSuccessType">
<sequence>
<element name="ResultSet" type="mpqf:OutputQueryType" minOccurs="0"/>
</sequence>
</extension>
</complexContent>
</complexType>
```

4.3.7.2.1.4 Semantics of Protocol Data Format

Semantics of the `RequestNamedEntityRequest`:

Name	*Definition*
`RequestNamedEntityRequest`	Protocol message sent from the client to the SP to request named entities of ontologies stored in SP's repositories which meet user conditions.
`RequestNamedEntityRequestType`	Top-level type for Recognize Entity Protocol request messages. `RequestNamedEntityRequestType` extends `mpegm:ProtocolRequestType`.
`InputQuery`	User criteria for the recognition of a named entity. It is of type `NamedEntityRequestInputQueryType`

Semantics of the `RequestNamedEntityResponse`:

Name	*Definition*
`RequestNamedEntityResponse`	Protocol message sent from the SP to the client with the results of the operation.
`RequestNamedEntityResponseType`	Top-level type for Request Named Entity

Name	*Definition*
	Protocol response messages. `RequestNamedEntityResponseType` extends `mpegm:ProtocolResponseType`.
`RequestNamedEntitySuccess`	Response in case of success. It is of type `RequestNamedEntitySuccessType`
`RequestNamedEntityFailure`	Response in case of failure.

Semantics of the `NamedEntityRequestInputQueryType`:

Name	*Definition*
`NamedEntityRequestInputQueryType`	`NamedEntityRequestInputQueryType` extends `mpqf:QueryType`.
`NamedEntityLabel`	A free text string entered by the user, which is used to match named entities.
`TargetNamespace`	A URI identifying the namespace within which the returned entities should be defined.
`EntityType`	A URI identifying the type of the requested entity. (Property, class, individual)

Semantics of the `RequestNamedEntitySuccessType`:

Name	*Definition*
`ResultSet`	The results of the operation. The result set contains the URI of the named entity that matches user's query and available description. It is of type `mpqf:OutputQueryType`.

4.3.8 Identify Content

4.3.8.1 Description

The Identify Content elementary service enables a User to obtain a unique identifier for a Versatile Digital Item or any of its components.

4.3.8.2 Protocol Specification

This elementary service follows the protocol specification of the MPEG-M Identify Content Elementary Service, defined in MPEG-M Part 4 [3].

4.3.9 Identify User

4.3.9.1 Description

The Identify User elementary service enables a User to obtain an identifier. This identifier is kept by the service provider, so that it can later be used for retrieving it by a third party (e.g. an application) using the user's credentials. If the user has already been identified, the Identify User SP will check the request to decide whether it will issue a new one, setting it to be the user's identifier when asked. Otherwise, it will return the existing identifier for the user.

4.3.9.2 Protocol Specification

This elementary service has the following protocol specification:

Steps	User	Service Provider
1.	A User sends an `IdentifyUserRequest` message. This message may include the public key of the User, digitally signed, and an indication whether a new identifier should be generated, in case there exists another one.	
2.		The Identify UserService Provider checks the validity of the user's public key and sends back an `IdentifyUserResponse` message, which, in case of success, contains the generated (or existing) identifier.

4.3.9.3 Syntax of Protocol Data Format

```
<!-- Definition of IdentifyUserRequest -->
<element name="IdentifyUserRequest" type="mpegm:IdentifyUserRequestType"/>
<complexType name="IdentifyUserRequestType">
<complexContent>
<extension base="mpegmb:ProtocolRequestType">
<sequence>
<element name="UserKey" type="dsig:KeyInfoType" minOccurs="0"/>
      <element name="RequestNewIdentifier" type="boolean" minOccurs="0"/>
</sequence>
</extension>
</complexContent>
</complexType>
</element>

<!-- Definition of IdentifyUserResponse -->
```

```
<element name="IdentifyUserResponse"
type="mpegm:IdentifyUserResponseType"/>
<complexType name="IdentifyUserResponseType">
<complexContent>
<extension base="mpegmb:ProtocolResponseType">
<choice>
<element name="IdentifyUserResponseSuccess"
type="mpegm:IdentifyUserResponseSuccessType"/>
<element name="IdentifyUserResponseFailure"
type="mpegmb:ProtocolFailureType"/>
</choice>
</extension>
</complexContent>
</complexType>
<complexType name="IdentifyUserResponseSuccessType">
<complexContent>
<extension base="mpegmb:ProtocolSuccessType">
<sequence>
<element name="UserIdentifier" type="mpeg7:UniqueIDType"/>
</sequence>
</extension>
</complexContent>
</complexType>
```

4.3.9.4 Semantics of Protocol Data Format

Semantics of `IdentifyUserRequest`:

Name	*Definition*
`IdentifyUserRequest`	The Request to Identify some User
`IdentifyUserRequestType`	Top-level type for `IdentifyUserRequest`, which extends the generic `ProtocolRequestType`.
`UserKey`	The digitally signed public key of the user (or even the whole certificate).
`RequestNewIdentifier`	Indicates whether a new identifier must be generated in case there is already one for this user.

Semantics of `IdentifyUserResponse`:

Name	*Definition*
`IdentifyUserResponse`	The Response of the service. If the user has been successfully identified, it will contain the user identifier.

`IdentifyUserResponseType`	Top-level type for `IdentifyUserResponse`, which extends the generic `ProtocolResponseType`.
`UserIdentifier`	A unique identifier issued to the user.
`IdentifyUserResponseSuccess`	Response in case of success.
`IdentifyUserResponseFailure`	Response in case of failure.

4.3.10 Inject Content

4.3.10.1 Description

The purpose of the Inject Content Elementary Service is to make a VDI discoverable via semantic search operations. The service may be called by users or activated automatically by the system when a new VDI is published.

It orchestrates calls to the following technology engines:

- Digital Item TE and Metadata TE: these TEs access and extract the metadata portion of the VDI and are part of the standard MPEG-M part 2.
- Overlay TE: this TE broadcasts metadata, VDI identifiers, expiry date and other VDI-level information (see D4.1), for VDIs of a given semantic type, to the overlay network of peers responsible for maintaining that specific semantic type. It proceeds to update local status tables and manages pointers so that the VDIs are discoverable by semantic search operations. The technology engine exposes a standard API but implements special algorithms to distribute metadata inside the semantic overlay. It will be further specified in D3.2. The design of the TE will be based on the basic architectural concepts specified in D3.1.

4.3.10.2 Protocol Specification

Steps	User	Service Provider
1.	A User sends an `InjectContentRequest` message. This message includes a reference to the VDI.	
2.		The Inject Content Service Provider replies with a Success/Failure message.

4.3.10.3 Syntax of Protocol Data Format

```
<!-- ################################################ -->
<!--  Inject Content                                  -->
<!-- ################################################ -->

<!-- Definition of InjectContentRequest -->
<element name="InjectContentRequest" type="conv:InjectContentRequestType"/>
<complexType name="InjectContentRequestType">
<complexContent>
<extension base="mpegm:ProtocolRequestType">
<sequence>
<element name="ContentEntity" type="mpegm:ContentEntityType"/>
      </sequence>
</extension>
</complexContent>
</complexType>

<!-- Definition of InjectContentResponse -->
<element name="InjectContentResponse"
type="conv:InjectContentResponseType"/>
<complexType name="InjectContentResponseType">
<extension base="mpegm:ProtocolResponseType">
<choice>
<element name="InjectContentSuccess" type="mpegm:ProtocolSuccessType"/>
<element name="InjectContentFailure" type="mpegm:ProtocolFailureType"/>
</choice>
</extension>
</complexType>
```

4.3.10.4 Semantics of Protocol Data Format

Semantics of the `InjectContentRequest`

Name	*Definition*
`InjectContentRequest`	The Request to Inject Content

Semantics of the `InjectContentResponse`

Name	*Definition*
`InjectContentResponse`	The Response indicating whether the Content has been successfully injected or not
`InjectContentSuccess`	Response in case of success
`InjectContentFailure`	Response in case of failure

4.3.11 Package Content

4.3.11.1 Description

The Package Content elementary service is used to prepare a VDI for delivery. That is, this service is responsible for creating a file out of the VDI, or generating a stream of VDI fragments, and binding it, to the transport protocol.

4.3.11.2 Protocol Specification

This elementary service follows the protocol specification of the MPEG-M Package Content Elementary Service, defined in MPEG-M Part 4 [3].

4.3.12 Present Content

4.3.12.1 Description

The Present Content elementary service enables a user to set the stylesheet for the presentation of a VDI. The service provider will apply the stylesheet to the VDI and return the corresponding XHTML document.

4.3.12.2 Protocol Specification

This elementary service has the following protocol specification:

Steps	User	Service Provider
1.	A User sends a `PresentContentRequest` message. This message includes the VDI, plus the transformations that are to be done or the CSS to be applied.	
2.		The Present ContentService Provider is able to display to the user contracts or parts thereof. Upon completion of the task, the SP sends back a message `PresentContentResponse` with the XHTML version of the presented content or a reference to a xml document where the CSS has been applied.

4.3.12.3 Syntax of Protocol Data Format

```
<!-- ################################################ -->
<!--  Present Content                                 -->
<!-- ################################################ -->

<!-- Definition of PackageContentRequest -->
<element name="PresentContentRequest"
type="conv:PresentContentRequestType"/>
<complexType name="PresentContentRequestType ">
<complexContent>
<extension base="mpegm:ProtocolRequestType">
<sequence>
<element name="ContentEntity" type="mpegm:ContentEntityType"/>
<choice>
<element name="xsl:transform" />
<element name="CSSStylesheet" type="anyURI" />
</choice>
</sequence>
</extension>
</complexContent>
</complexType>

<!-- Definition of PresentContentResponse -->
<element name="PresentContentResponse"
type="conv:PresentContentResponseType"/>
<complexType name="PresentContentResponseType">
<complexContent>
<extension base="mpegm:ProtocolResponseType">
<choice>
<element ref="xhtml:html"/>
<element name="ContentRepresentation" type="anyURI"/>
</choice>
</extension>
</complexContent>
</complexType>
```

4.3.12.4 Semantics of Protocol Data Format

Semantics of the `PresentContentRequest`

Name	*Definition*
`PresentContentRequest`	The Request to Present some Content
`ContentEntity`	A VDI or a reference to it
`CSSStylesheet`	A CSS document used to present the VDI.
`xsl:transform`	The XSL transformation to be applied in order to create an XHTML document out of the VDI.

Name	*Definition*
PresentContentResponse	The Response containing the Content in a presentable form
xhtml:html	The Content (VDI) provided at the request, formatted as an XHTML document
ContentRepresentation	A URI with an xml document containing the VDI and the CSS given as input.

4.3.13 Process Content

4.3.13.1 Description

The Process Content elementary service enables a user to modify the content of a VDI. The result will be a new VDI, following the semantics of the update VDI (i.e. keeping the same sequence identifier as its predecessor).

4.3.13.2 Protocol Specification

This elementary service follows the protocol specification of the Process Content elementary service, defined in MPEG-M Part 4 [3].

4.3.14 Process License

4.3.14.1 Description

The Process License elementary service enables a user to create a license, based on an already existing one. The new license may refer to different resources, different principals or maybe based on its predecessor template.

4.3.14.2 Protocol Specification

This elementary service follows the protocol specification of the Process License elementary service, defined in MPEG-M Part 4[3].

4.3.15 Request Content

4.3.15.1 Description

A user uses the Request Content elementary service to request permission to access a VDI.

4.3.15.2 Protocol Specification

This elementary service will follow the protocol specification for the MPEG-M Request Content elementary service, defined in MPEG-M Part 4 [3].

4.3.16 Request Event

4.3.16.1 Description

The Request Event elementary service enables a user to ask for an Event Report or an Event Report Request associated with a certain VDI.

4.3.16.2 Protocol Specification

This elementary service follows the protocol specification for the MPEG-M Request Event elementary service, defined in MPEG-M Part 4 [3].

4.3.17 Revoke Content

4.3.17.1 Description

The Revoke Content elementary service enables a user to revoke, i.e. to unpublish/unsubscribe a VDI. Revoke Content messages are delivered to the peers responsible for managing the semantic type of the VDI to be revoked. In order to revoke content, the user has to have the appropriate rights.

4.3.17.2 Protocol Specification

Steps	User	Service Provider
1.	A User sends a `RevokeContentRequest` message. This message includes a reference to the VDI and additionally, it may include a license, a revocation date and a revocation reason.	
2.		The RevokeContentService Provider replies with a Success/Failure message.

4.3.17.3 Syntax of Protocol Data Format

```
<!-- ################################################ -->
<!--  Revoke Content                 -->
<!-- ################################################ -->
```

```
<!-- Definition of RevokeContentRequest -->
<element name="RevokeContentRequest" type="conv:RevokeContentRequestType"/>
<complexType name="RevokeContentRequestType">
<complexContent>
<extension base="mpegm:ProtocolRequestType">
<sequence>
<element name="ContentEntity" type="mpegm:ContentEntityType"/>
<element name="DateOfRevocation" type="dateTime" minOccurs="0"/>
<element name="RevokeReason" type="RevokeReasonType"
minOccurs="0"/>
<element name="LicenseEntity" type="LicenseEntityType"
minOccurs=”0”/>
      </sequence>
</extension>
</complexContent>
</complexType>

<!-- Definition of RevokeContentResponse -->
<element name="RevokeContentResponse"
type="conv:RevokeContentResponseType"/>
<complexType name="RevokeContentResponseType">
<extension base="mpegm:ProtocolResponseType">
<choice>
<element name="RevokeContentSuccess" type="mpegm:ProtocolSuccessType"/>
<element name="RevokeContentFailure" type="mpegm:ProtocolFailureType"/>
</choice>
</extension>
</complexType>
```

4.3.17.4 Semantics of Protocol Data Format

Semantics of the `RevokeContentRequest`

Name	*Definition*
`RevokeContentRequest`	The Request to Revoke some Content
`ContentEntity`	A VDI or a reference to it
`DateOfRevocation`	The date that this Content has been revoked
`RevokeReason`	A `RevokeReasonType` indicating the reason this content has been revoked
`LicenseEntity`	A `LicenseEntityType` providing information on the rights of the user requesting to revoke the Content

Semantics of the `RevokeContentResponse`

Name	*Definition*
`RevokeContentResponse`	The Response indicating whether the Content

	has been revoked
`RevokeContentSuccess`	Response in case of success
`RevokeContentFailure`	Response in case of failure

4.3.18 Search Content

4.3.18.1 Description

The Search Content elementary service enables users to search for VDIs published in CONVERGENCE.

The search query may use:

- *unstructured criteria*. Free text description.
- *semi-structured criteria.* Property-value pairs where property is based on a model and value is free text.
- *structured criteria.* Property-value pairs where property and value are both based on a model.

In the latter two cases, the user may use the Describe Content elementary service to request entities of existing models and construct the criteria for the query.

4.3.18.2 Protocol Specification

This elementary service follows the protocol specification for the Search Content elementary service, defined in MPEG-M Part 4 [3].

4.3.19 Store Content

4.3.19.1 Description

The Store Content elementary service allows for the transfer and the storage of a VDI in a remote (or local) device.

4.3.19.2 Protocol Specification

This elementary service follows the protocol specification of the MPEG-M Store Content elementary service, defined in MPEG-M Part 4 [3].

4.3.20 Store Event

4.3.20.1 Description

The Store Event elementary service enables a user to store an Event Report Request or an Event Report that may occur on a device.

4.3.20.2 Protocol Specification

This elementary service follows the protocol specification of the Store Event elementary service, defined in MPEG-M Part 4 [3].

5 Network Protocols Specification

5.1 Scope of the specification

This section specifies the COnvergence NETwork (CONET) protocols and algorithms that support the CONET API defined in [5].

This deliverable specifies CONET protocols and algorithms used to retrieve named-data, i.e. to support the GET operation described in [5]. Future deliverables will describe additional functionality (e.g. the "send-to" API primitive), including delivery of un-named data, point-to-multipoint data distribution (for video streaming), and data-centric security.

We point out that the CONET protocols and algorithms described here will not be implemented in the phase 1, track 1 of the planned experimentation (see Description of Work), but only in the following phases, described in the Description of Work.

As shown in Figure 2, in phase 1, track 1, CONET functionality are made available to the middleware level by the middleware CONET Technology Engine (CONET TE), which is based on a plain TCP/IP network. The CONET TE is used by middleware Engines to perform "raw" CONET operations (i.e., Advertise, Get, SendToName, etc.), as specified in [5]. These operations enable a middleware Engine to *directly* exploit CONET functionality to handle named-resources. For instance, middleware Engines can use the CONET TE to store and advertise VDIs or any other named-resources, so as to make them accessible by remote middleware entities. Middleware Engines can also use the CONET TE to retrieve advertised named-resources.

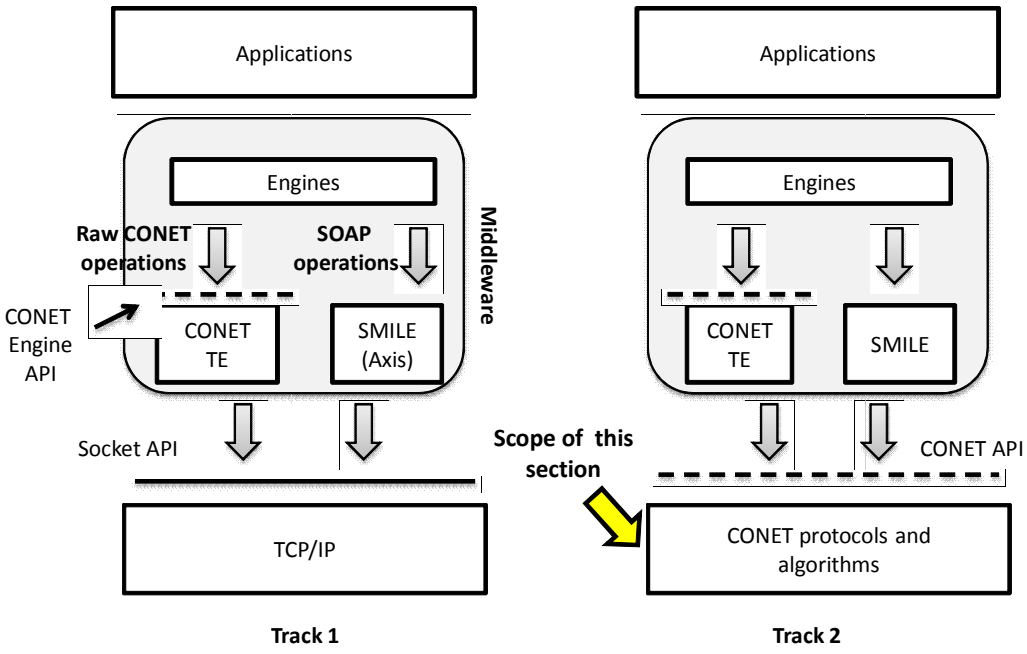


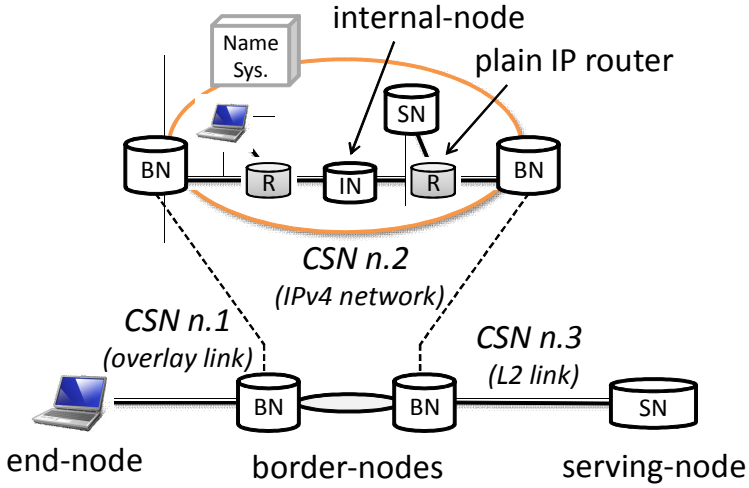
Figure 2: software architecture and scope of this specification section

Middleware Engines will use SOAP to exchange the XML messages defined in section 4.3. The SOAP implementation is provided by the SMILE framework [6], which in turn provides an Axis-like interface and uses the plain TCP/IP network.

In the subsequent experimentation phases in track 2, the CONET TE and the SMILE framework will be modified so as to interwork with the CONET protocols and algorithms specified in this deliverable. In this way, the migration from a TCP/IP to a CONET network will not require changes in the middleware and application software developed during phase 1, track 1.

5.2 Network Overview

As described in [5] and shown in Figure 3, CONET is an inter-networking architecture that interconnects underlying CONET Sub Networks (CSN), in the same way as IP interconnects sub-networks.



Path state:
[UDP/IP end-node, UDP/IP BN] – CSN n.1
[IP BN, IP BN] – CSN n.2
[MAC BN, MAC SN] - CSN n.3

Figure 3: CONET architecture

A CONET sub-network interconnects two or more CONET nodes, using an *under-CONET* technology to transfer CONET Information Units (CIUs) . Under-CONET technologies can be: point-to-point Layer 2 links (e.g. PPP), Layer 2 networks (e.g., Ethernet), overlay links (e.g. UDP over IP) or arbitrarily large IP networks, enriched with content-centric functionality. In this latter case we talk about IP-CSN (see Section 5.7.3).

CONET nodes are "logically" classified as end-nodes, serving-nodes, border-nodes (BNs) and internal-nodes (INs), depending on their function in the CONET.

End-nodes are devices that request named-resources and route-by-name the requests towards the best serving-nodes.

Serving-nodes are the original sources of named-resources.

Border-nodes are devices that: i) route-by-name requests of named-resources towards the best serving-nodes; ii) route back data toward the requesting end-nodes; iii) provide in-network caching functionality.

Finally, internal-nodes are extended IP routers that are content-aware and able to support in-network caching; these routers use plain IP mechanisms to forward CONET data units.

5.3 Protocol stack

Figure 4 describes the protocol stack below the CONET API.

The end-nodes implement transport-level functionality, supporting reliability and flow control, and providing the CONET API [5].

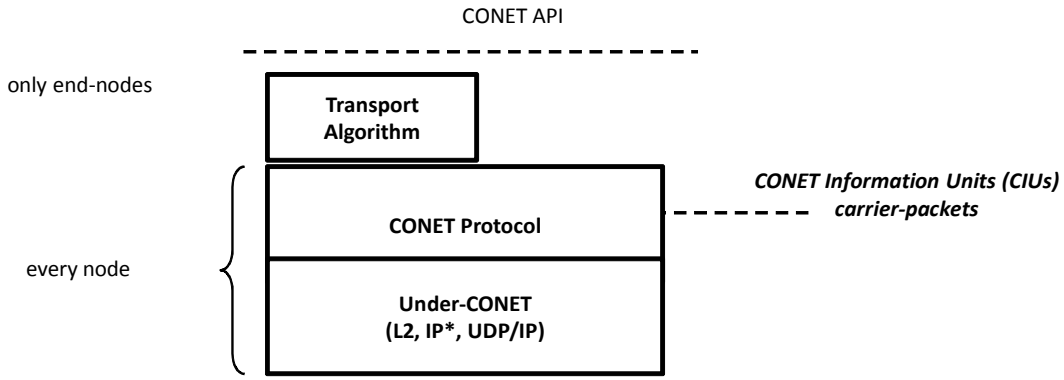


Figure 4: Protocol stack below the CONET API

Every CONET node implements the CONET and the *Under-CONET* protocols.

The CONET protocol is connectionless, handles CONET data-units and provides other specific functionality, like packetization, caching, security, routing by name and by path-info. The CONET protocol layer is divided into two sub-layers, whose data units are respectively denoted as "CONET Information Units" (CIU) and "carrier-packets".

An Under-CONET layer is any technology able to transfer a carrier-packet between two network or link end-points. Therefore, an Under-CONET technology could be either a link-layer, or network-layer or transport-layer technology. In the case of an IP network-layer technology, we enhance the IP layer so as to integrate part of CONET protocol functionality within IP (this is the reason why we use the tag IP* in Figure 4). To this end, we have defined a CONET IPv4 Option (see Section 5.7.3). In future deliverables we will also specify a CONET IPv6 Extension Header, so as to extend our solution to IPv6.

5.4 Network Identifier (NID)

The network-identifier (NID) uniquely identifies a named-resource at CONET level and has the form

```
NID = <namespace ID, name>
```

The `NID namespace ID` determines the structure of the rest of the NID, i.e. of the `name` field. Thus, the `name` field is a namespace-specific string. Up to now, we have defined two NID namespaces (see Table 1): one, derived from [7], uses variable length strings for the `name` field; the other one is based on fixed length hashes of flat-strings and has been developed within the CONVERGENCE project. The two namespaces are assigned the following NID name space IDs.

Table 1: Default CONET namespaces

Namespace ID	Reference
1	VLL (Variable Length Label) NID namespace
2	PLHB (Principal/Label Hash Based) NID namespace

5.4.1 VLL (Variable Length Label) NID namespace

In the VLL (Variable Length Label) CONET namespace, the `name` is simply the string representation of a named-resource. As described in [7], NIDs are hierarchically structured so that an individual name is composed of a number of components. An authority is needed to ensure the uniqueness of the NIDs. This approach is similar to the approach used to guarantee the uniqueness of DNS names in today's Internet.

In the CONET API, a NID of the VLL NID namespace can be inserted in the form of an URN [9], as shown below:

```
urn:conet:vll:name
```

Current WEB names can be easily mapped into such a NID format. For instance, the WEB address `www.ict-convergence.eu/documents/doc1.txt` could remain exactly the same:

```
name = www.ict-convergence.eu/documents/doc1.txt
```

5.4.2 PLHB (Principal/Label Hash Based) NID namespace

In the Principal/Label Hash Based CONET namespace, the `name` is the composition of two hash values:

```
name = <hash (Principal) , hash (Label)>
```

`Principal` and `Label` are strings. A `hash` function transforms them to a fixed number of bytes. The concept of Principal and Label was originally proposed in [8] but with a different name format.

A Principal is the owner of her named-resources and uses the `Principal` identifier whose hash is unique in its namespace. The principal identifier has the syntax of a DNS name, e.g. "www.foo.com". A central naming authority is needed to ensure the uniqueness of the `Hash(Principal)`, i.e. a Principal cannot be assigned if its hash collides with an already assigned hash.

Label is an identifier chosen by the principal to uniquely differentiate her named-resources. Label is a flat string, e.g. "text1/text2/text3.txt", whose hash is unique among the other label hashes provided by the same principal.

The `Hash(Principal)` is an 8 bytes hash of a string representing the `Principal`. The `Hash(Label)` is a 6 bytes hash of a string representing the `Label`. The `Hash` function is the MD5 function truncated so as to maintain the most significant bytes.

In the CONET API, a NID in the PLHB namespace can be inserted as the URN [9]:

```
urn:conet:plhb:Principal/Label
```

or simply

```
Principal/Label
```

Current WEB names could be easily mapped to such a NID. For instance, the WEB address `www.ict-convergence.eu/documents/doc1.txt` could remain exactly the same:

```
Principal = www.ict-convergence.eu
Label = documents/doc1.txt
```

5.5 Packetization process of named-data

Figure 5 shows the packetization process. A named-resource (e.g. a VDI, a file, etc.) is split into different parts, or *chunks*. Each chunk is packaged in a *named-data CIU*, whose control information contains the network-identifier, the chunk sequence number, temporal-data and the security-data [7]. 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 could support variable chunk sizes.

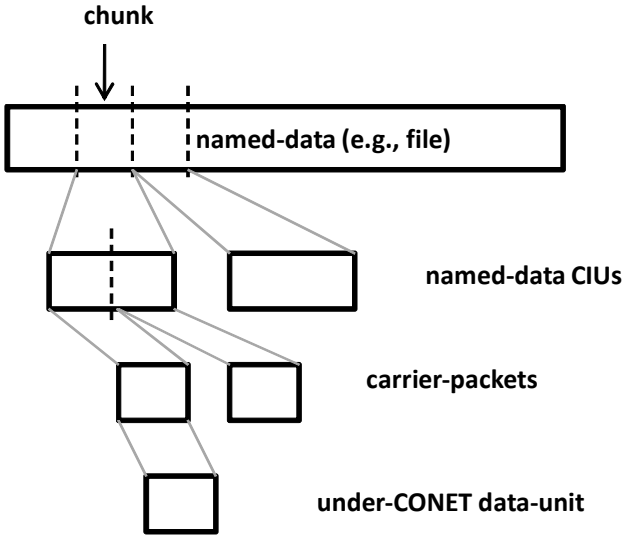


Figure 5: packetization process

A named-data CIU could be too large to be transported by a single under-CONET data-unit (e.g. Ethernet frame, or UDP datagram). Therefore, named-data CIUs, or parts of these CIUs, are segmented into a sequence of *carrier-packets*, whose size fits the payload capacity provided by the Under-CONET data-units. A carrier-packet includes all the necessary routing information, so as to be forwarded without requiring any kind of (slow) reassembly operation.

5.6 Data units

This section provides a preliminary definition of CONET data units. Future deliverables will provide a more thorough description, and the specification of other functionality such as delivery of un-named data (i.e. send-to API primitive), point-to-multipoint data distribution (for video streaming), and data-centric security.

To define our data units, we began with the proposals in [7][10] and went on modifying them in terms of notation and functionality. As concerns notation, the "interest packets" and "data packets" proposed in [7][10] correspond to our *interest CIU* and *named-data CIU*, respectively. However, their protocol information is different. In addition, we introduce the concept of carrier-packets. The new concept simplifies network forwarding, facilitating the provision of CONET functionality on high speed backbone nodes.

Following the approach adopted in [10], the data formats of the CONET Information Units (CIUs) are defined by XML schemas and encoded with explicitly identified field boundaries. This design permits field values of arbitrary length. The use of XML structures does not imply that field values are text strings nor does it require that messages be encoded as human-readable text. Most fields, including those that identify content, are defined to contain arbitrary binary values.

The format of carrier-packet complies with a bit-level description, as in classical network protocols, like IP, TCP, UDP, etc. This approach facilitates high speed forwarding of carrier-packets, eliminating the processing effort required to parse XML.

5.6.1 Named-data CIU

Figure 6 illustrates the structure of the named-data CIU; the related XML elements are described in Table 2 (the formal XML schema will be provided in future deliverables). A named-data CIU can be indentified and processed without using any external information. It is thus entirely self-contained.

A named-data CIU is a self-contained container used by the CONET protocol to package a chunk of a named-data.

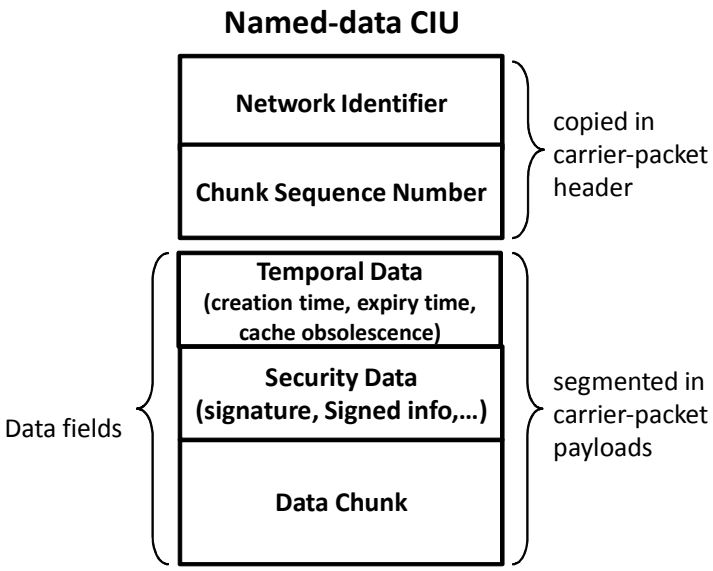


Figure 6: named-data CIU

A named-data CIU is uniquely identified within the CONET by the couple <NID, Chunk Sequence Number>. Furthermore, the validity of a named-data CIU can be checked by using information contained in the named-data CIU itself, i.e. it is self-authenticating.

Table 2: XML elements of the Named-data CIU

Field	Type	Meaning
Network Identifier	String	The NID of the named-data the named-data CIU refers to

Chunk Sequence Number	Integer	The number of the chunk of the named-data that is contained in the named-data CIU
Creation time	Integer	Creation time of the named-data CIU in Unix time format.
Expiration time	Integer	Expiration time (in Unix time format) of the named-data CIU. After this time, serving nodes and border or internal nodes shall not forward and delete the named-data CIU.
Caching Obsolescence Period	Integer	Number of seconds, since the named-data CIU was cached. After the period has elapsed, a cache shall verify the validity of the cached CIU by contacting a serving node. A value of 0 means “not-cachable”
Security Data	To be defined	Information that makes it possible to verify the validity of the named-data CIU. A named-data CIU is valid when the Data Chunk is really a chunk of the named-data identified by the couple <NID, Chunk Sequence Number>; i.e. the Data Chunk is not a fake content.
Data Chunk	Binary	A sequence of bytes of a named-data.

5.6.2 Interest CIU

Figure 7 summarizes the structure of the Interest CIU; the related XML elements are described in Table 3 (the formal XML schema will be provided in future deliverables).

An Interest CIU is a request for a subset of the Data fields of a named-data CIU.

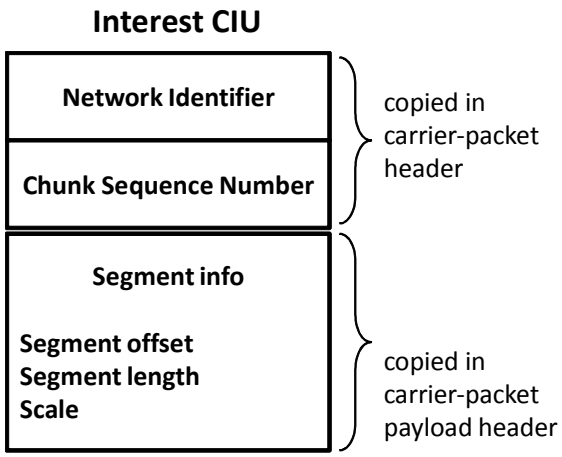


Figure 7: Interest CIU

For instance, an end-node could issue an interest CIU to request the block of bytes from byte 1000 to byte 2000 of the chunk n. 3 of a named-data with NID name = `www.ict-convergence.eu/documents/doc1.txt`. In this case, the CONET will send back the block of bytes 1000-2000 of the Data fields (i.e. Temporal-Data + Security-Data + Chunk Data, see Figure 6) of the named-data CIU identified by the couple <NID, Chunk Sequence Number>.

Table 3: XML elements of the Interest CIU

Field	Type	Meaning
Network Identifier	String	The NID of the named-resource addressed by the Interest CIU
Chunk Sequence Number (SN)	Integer	The number of the chunk of the named-resource addressed by the Interest CIU
Segment offset	Integer	Number of the first byte of the Data fields (see Figure 6) requested by the Interest CIU
Segment length	Integer	Length of the piece of byte of Data fields requested by the Interest CIU
Scale	Integer	Scaling factors of the Segment offset and Length fields

5.6.3 Carrier-packet

Figure 8 describes the structure of the carrier-packet; the related fields are described in Table 4.

Carrier-packets are low-level carriers of CIUs and are the data-units of the network forwarding process.

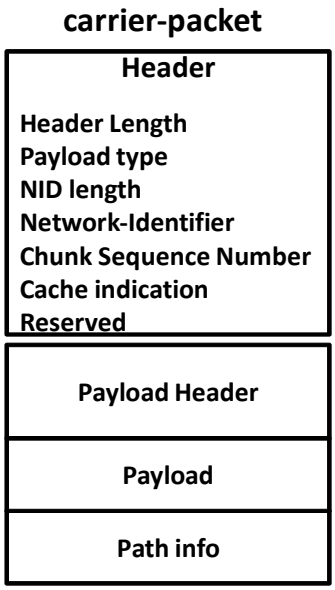


Figure 8: carrier-packet

We introduce carrier-packets because a named-data CIU could be too large to be transported by a single Under-CONET data-unit and thus needs to be segmented. A named-data CIU corresponds to a chunk of about 256/512 kbytes. However, the maximum data unit size of Under-CONET technologies can be smaller (e.g. 1500 bytes for Ethernet, 64 kbytes for UDP).

Carrier-packets are reassembled in border-nodes or in internal-nodes which are instructed to cache the related named-data CIU, and in end-nodes; validation of the content requires reassembly.

Carrier-packets are organized in four parts: *header*, *payloadheader*, *payload*, and *path-info* described in Table 4.

Table 4: fields of the carrier-packet

Field	Length (bits)	Meaning
Header		
Header length	8	Length of the carrier-packet header

Payload Type	4	Type of the Payload Value – Meaning 0 – Reserved 1 – bytes of a Interest CIU 2 – bytes of a Named-data CIU 2..15 – Reserved
NID Length	2	Code identifying the NID Length: Value – NID Length 0 – 16 bytes 1 – Reserved 2 – NID starts with a one byte length field (NID length in bytes) 3 – Reserved
Network Identifier	variable	The NID of the named-resource addressed by the carrier-packet
Chunk Sequence Number (SN)	variable	The chunk Sequence Number (SN) of the named-resource addressed by the carrier packet. The SN is coded with a variable number of bytes. An initial bit pattern determines the length of the SN field, i.e. we adopt a prefix-coding approach. Initial Bit pattern – SN Length 0 – 1 byte 10 – 2 bytes 110 – 3 bytes 1110 – 4 bytes 11110000 – 5 bytes 11110001 – 6 bytes Binary patterns from 11110010 to 11111111 are reserved. They can be used to extend the SN range if needed. With this option, we can have up to 2^40 chunks in a named-data. Assuming a relatively small chunk size of 1 kbytes, it is possible to have a named-data of 1099 TeraBytes; by assuming a

		more reasonable chunk size of 256 Kbyte it is possible to have a named-data of 281474 TeraBytes.
Cache indication	1	This bit field is used to control cache operations. Value – Meaning 0 – No cache 1 – Cache In the case of an Interest-CIU, if the bit is set to "No cache" it requests to nodes that are processing the CIU not to look for content in the cache, but to forward the request toward the source. In the case of a named-data CIU, if the bit is set to "No cache" it requests to nodes that are processing the CIU not to cache the content.
Reserved	1	A bit reserved for future use that aligns the carrier-packet header to a byte boundary.

Payload Header

Payload Header	variable	The information contained in the Payload Header is specific for each CIU type. The format of the Payload header field is reported below (5.6.3.1)

Payload

Payload	variable	The information contained in the payload is specific for each CIU type. The content of the Payload field is reported below (5.6.3.1

Path-info

Path-info	variable	The path-info control information is tightly associated with a specific communication session between an end-node and a serving-node (or an intermediate cache). The path-info field stores the end node address and the addresses of the set of border-nodes in the path between the end-node and the serving-node (or a border or internal node that provides a requested named-data CIU). The format of the path-info field is specified below (5.6.3.2)

5.6.3.1 Payload Header and Payload fields

5.6.3.1.1 Interest CIU

Table 5: Payload Header and Payload fields of the Interest CIU

Payload Header

Field	Length (bits)	Meaning
Segment offset	16	The same as for field of Interest CIU (see Table 3)
Segment length	16	The same as for Interest CIU (see Table 3)
Scale	8 (4+4)	The same as for Interest CIU (see Table 3)

Payload

Payload	0	void

5.6.3.1.2 Named-data CIU

Table 6: Payload Header and Payload fields of the named-data CIU

Payload Header

Field	Length (bits)	Meaning
Segment offset	16	Number of the first byte of Data fields (see Figure 6) contained in the payload
Segment length	16	Length of the piece of byte of Data fields contained in the payload
Scale	8 (4+4)	Scaling factors of the Segment offset and Length fields
More Chunk	1	Flag indicating more chunks beyond this one Value – Meaning 0 – this is the last chunk of the named-data 1 – this is not the last chunk of the named data
More Segments	1	Flag indicating no more data beyond this one Value – Meaning 0 – this carrier packet contains the last bit of the Data fields of the named-data CIU 1 – this carrier packet does not contain the last bit of the Data fields of the named-data CIU
Data field size	4	These bits indicate the length of the Data fields of the named-data CIU. Value – Length x – 2^x * 1kB for 0<=x<14 15 – undefined value

Payload

Payload	variable	Sequence of bytes that are a piece (segment) of the Data Fields of the named-data CIU (see Figure 6)

Table 7: Path-info field of a carrier-packet

Field	Length (bits)	Meaning
len	8 or 16	The length field specifies the length of the path-info field in bytes. If the first bit of the length field is 0, the remaining 7 bits of the first byte are used as a length field and the length field and the pointer field are both one byte long. In this case the maximum value of the length of the path-info field is 127 bits. If the first bit of the length field is 1, the length field and the pointer field are both two bytes long. In this case, the maximum value of the length of the path-info field is 32767.
pointer	8 or 16	The pointer field specifies the offset, starting from the start of the path-info field. This information is used by CONET nodes for routing purposes.
Ad-type #1	8	Type of the next address field. Each address is represented as a couple (ad-type, address), or as a triple (ad-type, ad-length, address) if the address type is of variable length. The ad-type field is of size one byte and possible values are: Value – Meaning 0 – Reserved 1 – Public IPv4 address (length is 4 bytes, no ad-length needed) 2 – Public Ipv6 address (length is 16 bytes, no ad-length needed)) 3 Ethernet address (length is 6 bytes, no ad-length needed)) 4-255 Reserved
Ad-length #1	optional	Length of the next address field

Address #1	variable	Network address (e.g. IP v4 address)
...	...	...
...	...	...
Ad-type #n	8	...
Ad-length #n	optional	...
Address #n	variable	...

5.7 Transport of Data Units through CONET Sub Networks (CSN)

As specified in Section 5.2, CONET nodes are connected by CONET Sub Networks (CSNs). These may consist of layer 2 network/link, of an IP network enriched with content-centric functionality, or of an overlay link. In what follows, we describe how to encapsulate the carrier-packets in each of these cases.

The deployment of a content-centric network on top of a link-layer or as an overlay-layer are known approaches used in clean-slate approaches (e.g. the FP7 PURSUIT project) or in overlay approaches (e.g. the FP7 SAIL and ALICANTE projects). However, the idea of integrating content-centric functionality "in" IP, so as to *make IP content-aware* is a novel concept proposed by the CONVERGENCE project. The proposed approach makes it possible to exploit existing IPv4 networks and current and future networks based on IPv6.

5.7.1 Layer 2 CSN (clean-slate approach)

In Layer 2 CSNs, carrier-packets are encapsulated in the payload of the layer-2 protocol data unit.

5.7.2 Overlay- CSN (overlay approach)

In an overlay-CSNs, the carrier-packet is encapsulated in the payload of a UDP/IP packet.

5.7.3 IP- CSN (integration approach) – The CONET IPv4 Option

In an IP-CSN, a carrier-packet is "integrated" in an IPv4 packet as follows. The header of the carrier-packet is conveyed by a novel **IPv4 CONET option**, detailed in [11] and depicted in Figure 9; the other fields of the carrier-packet are inserted in the IP payload. The *IP protocol number* of the IP header is a new number possibly to be allocated by IANA to indicate "CONET".

The CONET IPv4 option is parsed by CONET nodes belonging to the IP-CSN (IP routers enhanced with CONET functionality).

Other "plain" IP routers, on the end-to-end path, simply do not parse the CONET option.

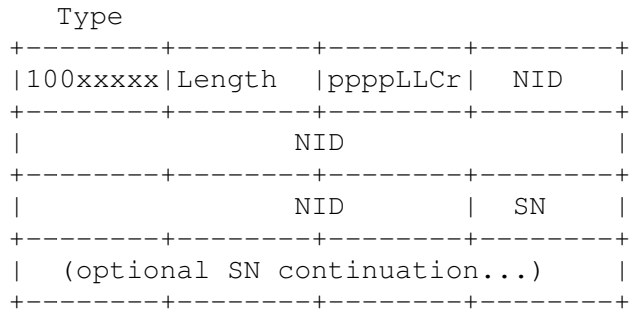


Figure 9: CONET IPv4 Option

Figure 9 shows the format of the CONET IPv4 option (see also [11]). The meaning of the fields is as follows.

The Type octet is viewed as having 3 fields: 1 bit copied flag, 2 bits option class, 5 bits option number. The related values are the following:

```
Copied flag:  1 (all IP fragments must carry the option)
Option class: 00 (control)
Option number: xxxxx (decimal) TO BE POSSIBLY ALLOCATED BY IANA
```

The Length field reports the variable length of the IP option in bytes.

pppp: CONET Information Unit Type. This four bits field is used to differentiate between different types of CONET Information Units (CIUs)

```
0     Reserved
1     Interest CONET Information Unit (Interest CIU)
2     Named-data CONET Information Unit (Named-data CIU)
2-15  Reserved
```

LL: NID Length Specification. This two bits field provides the length of the Network Identifier (NID) field or specifies how the NID length is provided:

```
0  16 bytes length
1  Reserved
2  NID starts with a one byte length field (NID length in bytes)
3  Reserved
```

C: cache indication. This one bit field is used to control cache operations.

```
0  No cache
1  Cache
```

(within Information Units that request a content (e.g. interest CIU), if the bit is set to "No cache" it indicates to the crossed nodes not to look for the content in the cache, but to forward

the request toward the source; within Information Units that carry content (e.g. named-data CIU), if the bit is set to "No cache" it indicates to the crossed nodes not to cache the content)

r : reserved. The two last bit of the first byte after the option length are reserved.

The other bits of the option are exactly the same as in the carrier-packet header described in Table 4.

In future deliverables, we will use the same approach to define a CONET IPv6 Extension Header.

5.8 Network operations

This section sketches some of the network operations involved in retrieving a named-data. Future deliverables will provide a detailed specification for these and other relevant operations.

In the examples described below, we consider a CONET network whose subnets are all IP-CSNs. We remind, however, that, the CONET architecture can also support the clean-slate and overlay scenarios.

5.8.1 Routing-by-name

The routing-by-name mechanism routes a CIU towards a named-resource. The routing-by-name process is performed in the end-node and in border-nodes that resolve a NID into the address of the CONET node representing the next hop towards the more convenient serving-node.

The basic procedure required for routing-by-name is `resolveNID`. This takes the NID as input and returns the address to which the incoming packet has to be forwarded over the Under-CONET layer.

```
resolveNid (NID) -> next hop address, output-interface
```

The `resolveNID` procedure is based on a name-based routing table. An entry of the name-based routing table contains the tuple:

```
<network-identifier, mask, next-hop address, output-interface>
```

This is similar to an entry in an IP routing table, but instead of net-prefixes we have *name-prefixes*, i.e. <network-identifier, mask> couples.

There are two possible approaches for populating a name-based routing table:

1- *Prefix-dissemination*: as suggested by [7], the BGP routing protocol could be used to disseminate name-prefixes, rather than net-prefixes.

2- *Lookup-and-cache*: this is a novel approach proposed by the CONVERGENCE project that makes it possible to limit the size of routing tables, and to ensure the scalability of the CONET architecture with respect to the number of addressed named-data. This approach envisages that a CONET node (end-node or border-node) should uses a fixed, limited number of rows in its name-based routing table; this table is meant to be used as a route cache. When a node misses the routing info required to route-by-name a data-unit, it looks up the related routing entry in a DNS like name-system and inserts the entry in the route cache. When all the rows are filled in, appropriate policies specify when new routing entries can substitute old ones. From a logical point of view, a name-system serves a single CONET subnet and a specific CONET namespace.

The prefix-dissemination and lookup-and-cache approaches can work independently. They can also be suitably combined, e.g. using prefix-dissemination for the most popular named-resources and lookup-and-cache for the remaining ones. The protocols involved will be specified in future deliverables.

5.8.2 Source-routing

The source-routing mechanism routes a CIU toward a specific network node (rather than toward a named-resource, as in the case of the routing-by-name mechanism).

The source-routing process is performed in the end-node, in the serving-node and in border-nodes. A serving node exploits source-routing to send back a named-data CIU to the requesting end-node. An end-node exploits source-routing to send a CIU to a "specific" network node.

Source-routing is supported by the path-info control information contained in the carrier-packets. Using path-info control information, a CONET node (end, serving or border) determines the address of the CONET node representing the next hop towards the destination node.

5.8.3 Retrieval of a piece of a named-data CIU

The retrieval of a piece of a named-data CIU involves a *request phase* and a *delivery phase*.

For instance, an end-node can request the block of bytes from byte 1000 to byte 2000 of the Data fields of chunk n. 3 of the named-data with NID name = `www.ict-convergence.eu/documents/doc1.txt`.

In the request phase, a request (Interest CIU) from an end-node is routed-by-name towards the most convenient serving-node that can provide the desired named-data CIU.

In the delivery phase, the requested block of bytes is routed back to the end node from the serving-node or from any cache available on the end-to-end path related to the desired named-data CIU. This reverse routing is accomplished by using a source-routing paradigm, based on the path-info field contained in the carrier-packets. Indeed, unlike the stateful approaches proposed by the CCNx [7][10] and EU FP7 COMET projects [13], CONVERGENCE proposes a *stateless content delivery* approach; CONET nodes do not maintain any information on the ongoing communication session and we believe that this approach improves the scalability performance of the CONVERGENCE network.

In what follows, we describe the operations performed by the different CONET nodes during the two phases.

5.8.3.1 Request phase

5.8.3.1.1 Processing in the end-node

An end-node that wants to retrieve a piece of a named-data issues an Interest CIU, which is transported by a carrier-packet, hereafter an *interest-carrier-packet*.

The end-node stores its IP address in the path-info field of the interest-carrier-packet and initializes the pointer field.

The interest-carrier-packet is encapsulated in an IP packet, as described in Section 5.7.3. To this end, the end-node has to determine the destination IP address for the interest-carrier-packet. The end-node performs a routing-by-name, trying to associate the NID with a next hop (i.e. with the IP address of the next hop). The next hop can be the serving-node (if the serving-node is in the same CONET subnet as the end-node) or a border-node of the CONET subnet (if the serving-node is in a different CONET subnet).

5.8.3.1.2 Processing in the serving-node

The serving-node receives the interest-carrier-packet in an IP packet addressed to itself, with an IP protocol number corresponding to "CONET"; the packet is internally dispatched to the "CONET entity" in the serving-node. The CONET entity recognizes that the carried CIU is an Interest CIU and responds by packaging the requested bytes in a novel carrier-packet, hereafter a *data-carrier-packet*. The path-info field of the interest-carrier-packet is piggybacked in the data-carrier-packet, so as to route back the packet to the requesting end-node.

5.8.3.1.3 Processing in the border-node

If the end-node and the serving-node are located in different CONET subnets, the interest-carrier-packet traverses one or more intermediate border-nodes.

A border-node will receive an interest-carrier-packet within an IP packet directed to itself, with an IP protocol number corresponding to "CONET". The packet is then internally dispatched to the CONET entity in the border-node. The CONET entity recognizes that the received packet is an interest CIU. It reads the NID and Chunk Sequence Number and determines which piece of the named-data CIU it needs to provide.

If the Cache field of the interest-carrier-packet is set to "No Cache" or if the field is set to "Cache" but the related named-data CIU is not available in the (optional) cache of the border-node, the border-node starts the routing-by-name process. It resolves the next hop of the interest-carrier-packet, which can be a serving-node in a different CONET subnet connected to the border-node, or another border-node in the path toward the serving-node. Before sending out the interest-carrier-packet, the border node adds its IP address in the path-info field and updates the pointer field.

If the Cache field of the interest-carrier-packet is set to "Cache" and the border-node has the requested named-data CIU in its cache, the border-node will reply to the interest CIU by packaging the requested bytes into a novel carrier-packet, hereafter a *data-carrier-packet*. The path-info field of the interest-carrier-packet is piggybacked in the data-carrier-packet so as to route back the data to the requesting end-node.

5.8.3.1.4 Processing in the internal-node

In the path between end-node and serving-node, the interest-carrier-packet can traverse several internal-nodes i.e. IP routers aware of the CONET IP Option. These nodes may have already cached the named-data CIU requested by the interest-carrier-packet.

The internal-node works as follows. When processing the IP header for a received IP packet, it finds that the packet contains the CONET IP Option.

If the Cache field of the Option (i.e. of the interest-carrier-packet header) is set to "No Cache", the internal-node forwards the packet using the destination IP address.

If the Cache field is set to "Cache", the internal-node hat to check the presence of the named-data CIU in its cache before forwarding the IP packet. It then reads the NID and Chunk Sequence Number in the CONET IP option and checks if the named-data CIU is in its cache.

If the named-data CIU is not present, it performs the normal IP processing.

If the named-data CIU is in the cache, the internal-node will process the remaining CONET protocol information carried in the IP payload and will reply to the interest CIU by packaging the requested bytes in a novel carrier-packet, hereafter the *data-carrier-packet*. The path-info field of the interest-carrier-packet is piggybacked in the data-carrier-packet, so as to route the packet back to the requesting end-node.

5.8.3.1.5 Processing in the legacy routers

In the path between end-node and serving-node, the interest-carrier-packet may traverse several legacy IP routers. Such routers will simply forward the packet by looking at the IP

destination address. Note that a requirement for such legacy routers is that they have to be configured not to drop IP packets carrying unidentified IP options; this is the current configuration in most routers (see [12]).

5.8.3.2 Delivery phase

5.8.3.2.1 Processing in the responding node

The responding node is the node that is able to provide the named-data CIU (identified by NID and Chunk Sequence Number) to a requesting end-node. It may be a serving-node that provides an original copy of the named-data CIU, or a border/internal-node that has a cached copy of the named-data CIU.

The responding node packages the requested bytes in a novel carrier-packet (the *data-carrier-packet*), copy the path-info of the interest-carrier-packet in the same field of the data-carrier-packet and integrates it in an outgoing IP packet [1], as specified in Section 5.7.3.

The related IP destination address is derived from the path-info information. The receiving node reads the last address in the list (the "CONET previous hop") and uses it as an IP destination address. It then updates the pointer field so that the next node uses the previous address in the path-info list as the CONET previous hop.

5.8.3.2.2 Processing in a border-node

When the end-node and the responding node are located in different CONET subnets, the data-carrier-packet traverses a set of intermediate border-nodes.

When a border-node receives a data-carrier-packet, the packet is contained in an IP packet directed to itself, whose IP protocol number corresponds to "CONET". The packet is then internally dispatched to the "CONET entity" in the border-node. The CONET entity reads the CONET IP option and recognizes that the received packet is a data-carrier-packet.

The border node reads the path-info field and uses the pointer field to identify the next “CONET previous hop” in the path towards the requesting end-node. The border-node forwards the data-carrier-packet in an outgoing IP packet, whose destination IP address will be set to the CONET previous hop retrieved from the path-info field. Before sending out the data-carrier-packet, it will shift the pointer field by one position.

If the Cache field of the data-carrier-packet is set to "Cache", the border node may choose to store the piece of the named-data CIU carried by the data-carrier-packet; however, it can directly reply to an interest-carrier-packet requesting those data only when it has reassembled the entire named-data CIU. Indeed, only when the CIU has been fully reassembled, the border node can verify the validity of the content, and avoid sending fake content.

[1]We assume that the requested bytes can be carried by a single data-carrier-packet; otherwise more carrier-packets will be sent sequentially.

5.8.3.2.3 Processing in an internal-node

Like a border-node, an internal-node may decide to cache a piece of the named-data CIU transported by a data-carrier-packet.

The internal-node receives an IP packet with an IP destination different from its own address and uses IP routing to immediately forward the packet.

If the Cache indication bit in the IP option is set to "Cache", the internal-node may also choose to store the piece of the named-data CIU. However, it will only be able to reply to a related interest-carrier-packet only when the entire named-data CIU has been reassembled, allowing it to verify the validity of the content and to avoid sending fake content.

5.8.3.2.4 Processing in the legacy routers

When a legacy router receives the data-carrier-packet, it simply forwards the packet on the base of its IP destination address. Note that the router has to be configured not to drop IP packets carrying unidentified IP options; this is the current configuration in most routers (see [12]).

5.8.4 Complete retrieval of a named-data – Transport algorithm

The complete retrieval of a named-data, i.e. a file with NID name = `www.ict-convergence.eu/documents/doc1.txt` involves a sequence of retrievals for all the pieces of named-data CIUs that form the complete named-data (see Figure 5).

This sequential retrieval process is handled by a *transport algorithm,* which also controls the rate at which single pieces are retrieved, (i.e. the rate at which the Interest CIUs are send out by the end-node), and ensures reliable transfer by requesting retransmission of missing data (see Figure 4).

The specification of the transport algorithm is left for future deliverables. However, we can already offer a general description of our idea, which is based on the receiver-driven, TCP-like approach proposed in [7].

The transport algorithm issues a sequence of Interest CIUs each of which requests only a small part of the whole named-data, 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 outstanding interest CIUs in the network.

6 Bibliography

[1] Study of ISO/IEC CD 23006-1 2nd Edition Multimedia Service Platform Technologies: Architecture, X. Wang, C. Timmerer, W. Lee, P. Kudumakis, January 2011. http://mpeg.chiariglione.org/working_documents/mpeg-m/pt1.zip.

[2] Study text of ISO/IEC CD 23006-2 2nd edition APIs, S. Matone, C. Timmerer, M. Grafl, W. Lee, A. Difino, January 2011. http://mpeg.chiariglione.org/working_documents/mpeg-m/pt2.zip.

[3] Study of ISO/IEC CD 23006-4 2nd edition Elementary Services, M. Grafl, S. Matone, M. Choi, V. Rodriguez, J. Delgado, and F. Chiariglione, January 2011. http://mpeg.chiariglione.org/working_documents/mpeg-m/pt4.zip.

[4] ISO/IEC 21000-5:2004, Information technology - Multimedia framework (MPEG-21) - Part 5: Rights Expression Language.

[5] CONVERGENCE Project Deliverable D3.2 (see also CONVERGENCE Project Deliverable D3.1).

[6] S. Salsano et al. “SMILE (Simple Middleware Independent LayEr) and its bindings: SMILE-JS (JSON over SIP),SMILE-JSH (JSON over SIP or HTTP)”, available at http://netgroup.uniroma2.it/SMS/TR/tr-smile.pdf.

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

[8] T. Koponen, M. Chawla, B.G. Chun, A. Ermolinskiy, Kye Hyun Kim, S. Shenker, I. Stoica: “A data-oriented (and beyond) network architecture”, Proc. of ACM SIGCOMM 2007.

[9] “URN Syntax” IETF 2141, RFC http://www.ietf.org/rfc/rfc2141.txt

[10] CCNx project website, http://www.ccnx.org.

[11] A. Detti et al., “An IPv4 Option to support Content Networking”, Internet Draft, draft-detti-conet-ip-option-00, Work in progress, March 2011.

[12] A. Detti, N. Blefari-Melazzi, S. Salsano, M. Pomposini: “CONET: A Content Centric Inter-Networking Architecture”, ACM SIGCOMM 2011, Workshop on Information-Centric Networking, 19 August 2011, Toronto, Ontario, Canada.

[13] W. K. Chai, N. Wang, I. Psaras, G. Pavlou, C. Wang, G. G. de Blas, F. J. Salguero, L. Liang, S. Spirou, A. Beben and E. Hadjioannou, "CURLING: Content-Ubiquitous Resolution and Delivery Infrastructure for Next Generation Services", IEEE Communications Magazine, Special Issue on Future Media Internet. To appear in the March 2011 issue.