| | |
|---|---|
| **Project Number:** | **FP7-257123** |
| **Project Title:** | **CONVERGENCE** |
| **Deliverable Type (Nature):** | **Prototype** |
| **Dissemination Level** | **Public** |
| **Deliverable Number:** | **D5.2** |
| **Contractual Date of Delivery to the CEC:** | **31.05.2012** |
| **Actual Date of Delivery to the CEC:** | **27.06.2012** |
| **Title of Deliverable:** | **Intermediate protocol architecture** |
| **Workpackage contributing to the Deliverable:** | **WP5** |
| **Editor:** | **Angelos-Christos Anadiotis, Charalampos Patrikakis, Iakovos Venieris, Richard Walker** |
| **Author(s):** | **Angelos-Christos Anadiotis (ICCS), Aziz Mousas (ICCS), Dimitra Kaklamani (ICCS), Georgios Lioudakis (ICCS), Konstantinos Papadopoulos (ICCS), Nikolaos Dellas (ICCS), Vasso Giotopoulou (ICCS), Thomas Huebner (MORPHO), Amit Shrestha (MORPHO), Mihai Tanase (UTI), Andrea Detti, Matteo Pomposini, Nicola Blefari Melazzi (CNIT)** |
| **Abstract:** | **D5.2 is classified in the DoW as a prototype. The D5.2 prototype comprises: i) the CONVERGENCE middleware protocols and technology engines APIs, implemented as JAVA interfaces and Web Services in the CONVERGENCE Middleware and in MXM, and ii) the CONVERGENCE network protocols and APIs.**<br>**Following the completion of the D5.2 prototype, the consortium decided to issue this report, considered complementary to the prototype.**<br>**The report, bearing the same name as the prototype, provides: i) a description of the CONVERGENCE user** |

| | |
|---|---|
| | **registration/authentication and licensing/authorization schemes; ii) a description of the CONVERGENCE middleware protocols and technology engines (some of which are included in MPEG-M, while others are CONVERGENCE-specific); iii) a description of the network protocols, and specifically of the CONET support for video streaming services.**<br><br>**These services provide an example of exploitation of CONET protocols and APIs and are included in the prototype and in a demo shown in relevant public events.** |
| **Keyword List:** | **Protocols, Middleware, Network, Engines, Authentication, Authorization, Video Streaming** |

# Executive Summary

D5.2 is classified in the DoW as a prototype. The D5.2 prototype comprises: i) the CONVERGENCE middleware protocols and technology engines APIs, implemented as JAVA interfaces and Web Services in the CONVERGENCE Middleware and in MXM, and ii) the CONVERGENCE network protocols and APIs.

The D5.2 prototype, based on the CONVERGENCE requirements and on the overall CONVERGENCE system protocol architecture, incorporates modifications and enhancements of the initial architecture, introduced in the course of the project.

Following the completion of the D5.2 prototype (which is an intermediate step towards the final D5.3), the consortium decided to issue this report, considered complementary to the prototype.

The report, bearing the same name as the prototype, provides: i) a description of the CONVERGENCE user registration/authentication and licensing/authorization schemes; ii) a description of the CONVERGENCE middleware protocols and technology engines (some of which are included in MPEG-M, while others are CONVERGENCE-specific); iii) a description of the network protocols and specifically of the CONET support for video streaming services. These services provide an example of exploitation of CONET protocols and APIs and are included in the prototype and in a demo shown in relevant public events.

CONVERGENCE specific engines and protocols are described in full. For engines and protocols included in the MPEG standard we offer references to the appropriate documentation.

# INDEX

# 1 Introduction

D5.2 is classified in the DoW as a prototype. The D5.2 prototype comprises: i) the CONVERGENCE middleware protocols and technology engines APIs, implemented as JAVA interfaces and Web Services in the CONVERGENCE Middleware and in MXM, and ii) the CONVERGENCE network protocols and APIs.

The D5.2 prototype, based on the CONVERGENCE requirements and on the overall CONVERGENCE system protocol architecture, incorporates modifications and enhancements of the initial architecture, introduced in the course of the project.

Following the completion of the D5.2 prototype (which is an intermediate step towards the final D5.3), the consortium decided to issue this report, considered complementary to the prototype.

The report, bearing the same name as the prototype, provides: i) a description of the CONVERGENCE user registration/authentication and licensing/authorization schemes; ii) a description of the CONVERGENCE middleware protocols and technology engines (some of which are included in MPEG-M, while others are CONVERGENCE-specific); iii) a description of the network protocols and specifically of the CONET support for video streaming services. These services provide an example of exploitation of CONET protocols and APIs and are included in the prototype and in a demo shown in relevant public events.

CONVERGENCE specific engines and protocols are described in full. For engines and protocols included in the MPEG standard we offer references to the appropriate documentation.

# 2 Terms and Definitions

| Term | Definition |
|---|---|
| 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. |
| Community Dictionary Service (CDS) | A CoMid Technology Engine that provides all the matching concepts in a user's subscription, search request and publication. |
| CONVERGENCE Device | A combination of hardware and software or a software instance that allows a user to access Convergence functionalities |
| CONVERGENCE Engine | A collection of technologies assembled to deliver specific functionality and made available to Applications and to other Engines via an API |
| CONVERGENCE Middleware level (CoMid) | The level of the CONVERGENCE architecture that provides the means to handle VDIs and their components. |
| CONVERGENCE Network (CoNet) | The Content Centric component of the CONVERGENCE Computing Platform level. The CoNet provides access to named-resources on a public or private network infrastructure. |
| CONVERGENCE System | A system consisting of a set of interconnected devices - peers and nodes - connected to each other built by using the technologies specified or adopted by the CONVERGENCE specification. See "Node" and "Peer". |
| Digital Item (DI) | A structured digital object with a standard representation, identification and metadata. A DI consists of resource, resource and context related metadata, and structure. The structure is given by a Digital Item Declaration (DID) that links resource and metadata. |
| Elementary Service (ES) | The most basic service functionality offered by the CoMid. |
| Entity | An object, e.g. VDIs, resources, devices, events, group, licenses/contracts, services and users, that an Elementary Service can act upon or with which it can interact. |
| ICN | Information Centric Network |
| Identifier | A unique signifier assigned to a VDI or components of a VDI. |

| | |
|---|---|
| License | A machine-readable expression of Operations that may be executed by a Principal. |
| Metadata | Data describing a resource, including but not limited to provenance, classification, expiry date etc. |
| MPEG eXtensible Middleware (MXM) | A standard Middleware specifying a set of Application Programming Interfaces (APIs) so that MXM Applications executing on an MXM Device can access the standard multimedia technologies contained in the Middleware as MXM Engines. |
| MPEG-M | An emerging ISO/IEC standard that includes the previous MXM standard. |
| Named service access point | A kind of named-resource, consisting of a service access point identified by a name. A named-service-access-point is a network endpoint identified by its name rather than by the Internet port numbering mechanism. |
| Named-data | A named-resource consisting of data. |
| Network Identifier (NID) | An identifier identifying a named resource in the CONVERGENCE Network. If the named resource is a VDI or an identified VDI component, its NID may be derived from the Identifier (see "Identifier"). |
| Policy routing | In the context of IP networks, a collection of tools for forwarding and routing data packets based on policies defined by network administrators. |
| Principal (CoNet) | The user who is granted the right to use a *CoNet Principal Identifier* for naming its named resources.<br>For example, the principal could be the provider of a service, the publisher or the author of a book, the controller of a traffic lights infrastructure, or, in general, the publisher of a VDI.<br>A Principal may have several Principal Identifiers in the CoNet. |
| Principal (Rights Expression Language) | The User to whom Permissions are Granted in a License. |
| Principal Identifier (CoNet) | The Principal identifier is a string that is used in the Network Identifiers (NID) of a CoNet resource, when the NID has the form:<br>NID = <namespace ID, hash (Principal Identifier), hash (Label)><br>In this approach, hash (Principal Identifier) must be unique in the namespace ID, and Label is a string chosen by the principal in such a way that hash (Label) is unique for in the context of the Principal Identifier. |

| | |
|---|---|
| Publish | The act of informing an identified subset of users of the CONVERGENCE System that a VDI is available. |
| Publisher | A user of CONVERGENCE who performs the act of publishing. |
| Publish-subscribe model | CONVERGENCE uses a content-based approach for the publish-subscribe model, in which notifications about VDIs are delivered to a subscriber only if the metadata / content of those VDIs match constraints defined by the subscriber in his Subscription VDI. |
| REL | Rights Expression Language |
| Resource | A virtual or physical object or service referenced by a VDI, e.g. media, Real World Objects, persons, internet services. |
| Scope (in the context of routing) | In the context of advertising and routing, the geographical or administrative domain on which a network function operates (e.g. a well defined section of the network - a campus, a shopping mall, an airport -, or to a subset of nodes that receives advertisements from a Service Provider). |
| Search | The act through which a user requests a list of VDIs meeting a set of search criteria (e.g. specific key value pairs in the metadata, key words, free text etc.). |
| SP | Service Provider |
| Subscribe | The act whereby a user requests notification every time another user publishes or updates a VDI that satisfies the subscription criteria defined by the former user (key value pairs in the metadata, free text, key words etc.). |
| Subscriber | A user of CONVERGENCE who performs the act of subscribing. |
| Tool | Software providing a specific functionality that can be re-used in several applications. |
| 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). |
| User Profile | A description of the attributes and credentials of a user of the CONVERGENCE System. |
| Versatile Digital Item (VDI) | A structured, hierarchically organized, digital object containing one or more resources and metadata, including a declaration of the parts that make up the VDI and the links between them. |

# 3 Middleware Protocol Architecture

## 3.1 General Middleware Architecture

The CONVERGENCE middleware consists of Protocol and Technology Engines, as shown in Figure 1. The former are used to call Elementary Services (one Elementary Service for each Protocol Engine), the latter provide the necessary technology support. Apart from these two fundamental types of engine, there also exists a set of Orchestrator Engines, whose role is to orchestrate complex operations using either a single Technology Engine or multiple technology engines, when they work towards a specific goal. Elementary Services often have to perform tasks that go beyond the functionality made possible by the Technology Engine APIs. As a result, each Elementary Service has its own Orchestrator Engine.

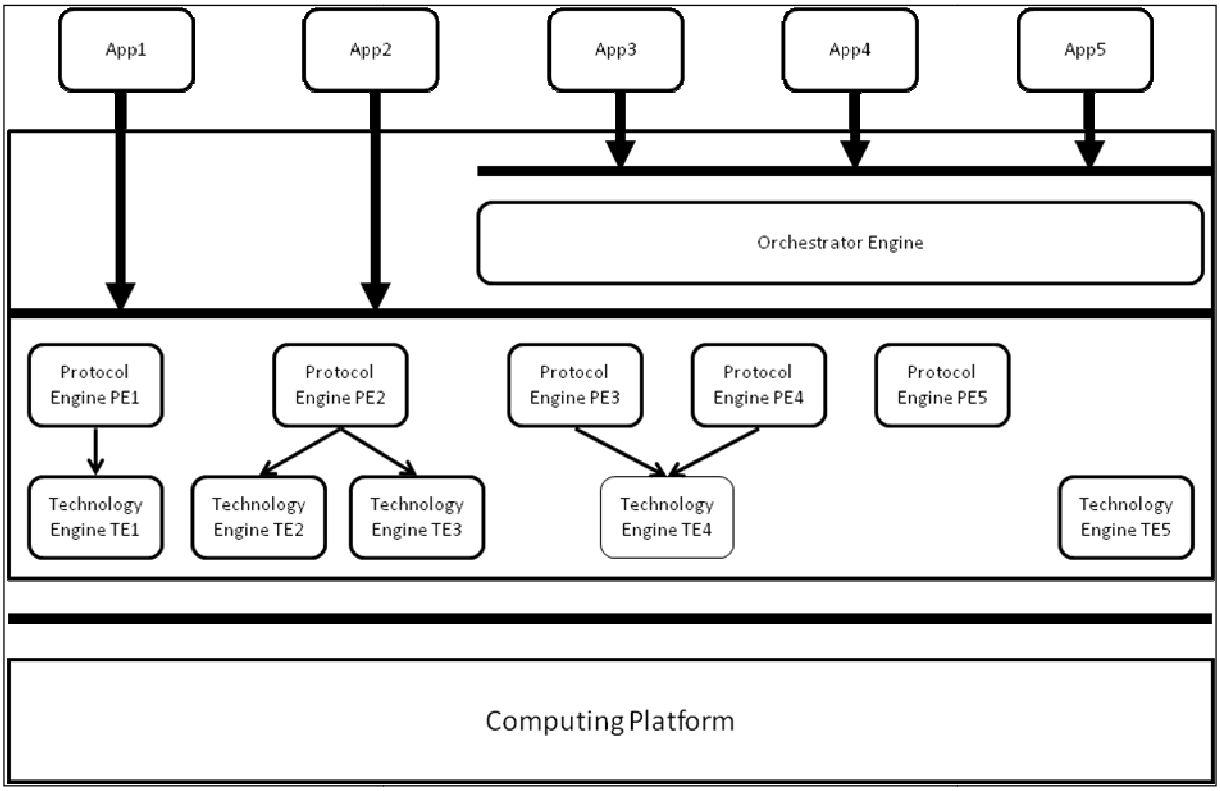


***Figure 1: CONVERGENCE Middleware Architecture***

## 3.2 User Registration and Authentication Scheme

Following the MPEG-M standard specification, user authentication in CONVERGENCE relies on the Security Assertion Markup Language (SAML) specification [20]. SAML does not specify specific mechanisms for verifying the users' identity; rather it assumes that the security context (user's identity verification) is established by external means/ protocols/technologies, and the SAML protocols are used to pass information about this security context, such as user's attributes, validity period, etc. SAML also provides Single-

Sign-On (SSO) capabilities, where the Identity Providers (IdP), verifying the users' credentials, may or may not be part of the CONVERGENCE ecosystem.

In CONVERGENCE, the role of the IdP is usually played by an IdentifyUser Service Provider where users can register by presenting their public key and obtaining an identifier. Every time a registered user requests to be authenticated, CONVERGENCE verifies the user's identity by verifying the public key used to obtain the user identifier and issues the appropriate SAML assertion.

However, interoperability requires that CONVERGENCE should also support non-CONVERGENCE identity providers, delegating the verification of users' credentials verification to trusted third parties, such as Google or Facebook. To achieve this, CONVERGENCE supports the OAuth 2.0 protocol[1] allowing users to access applications using their Google or Facebook account, without forcing them to use CONVERGENCE identity and authentication services.

### 3.2.1 User Registration

User registration to CONVERGENCE uses the Identify User Elementary Service[2]. To obtain an identifier the user uses a digital certificate or a public key. The identifier is stored in a secure repository in the client device (e.g. smart card). In this way, the identifier can be used to hide the user's real identity to applications. At the same time, the identifier can still be linked back to the identity of the real user via the Identify User Service Provider (always assuming the provider is trusted).

### 3.2.2 User Authentication

Username/password authentication schemes require the user to provide appropriate credentials. Only when these have been verified does the Authenticate User Elementary Service provide a SAML assertion to be stored in the client device. Key based schemes also require interactions between the client device and the Identify User service. In this case, the user identifier can be stored on a smart card, together with the private key whose public key counterpart was used to issue the identifier. When the user accesses the authentication application, the application sends a random challenge encrypted with the user's public key. This challenge is decrypted on the client side and sent back to the authentication application and encrypted with the applications' public key of the latter and the identifier for the user. The authentication application checks whether the identifier matches the user's public key. If so it triggers the Authenticate User Elementary Service, which provides a signed SAML assertion verifying the user's identity. This assertion is stored in the client device, as in the previous case.

---

[1] http://oauth.net/2/

[2] in the OAuth case, CONVERGENCE registration may not be required. All that is necessary is that the Authenticate User Elementary Service trusts the third party Identity Provider).

## 3.3 Licensing and Authorization Scheme

CONVERGENCE middleware is aligned with the MPEG-21 and the corresponding MPEG-M Rights Expression Languages (REL) [8][18], which it uses to specify and enforce access control rules for resources referenced by VDIs. The licenses created by the middleware can grant a set of rights, as specified in the MPEG-21 Rights Data Dictionary (RDD) [19], either to a specific user or a group. Groups are also represented by licenses. Thus, each user who is a member of this group must have a corresponding license.

The fundamental protocols supporting the licensing and authorization schemes are specified by the MPEG-M CreateLicense and AuthorizeUser Elementary Services. The CreateLicense protocol is used to create a license stating the rights of a particular user or group over a given resource under specified conditions. The AuthorizeUser protocol is responsible for verifying that a principal is allowed to access the resource protected by a license and, if that resource is encrypted, to provide the appropriate decryption key. Given that responsibility for access control is delegated to the AuthorizeUser Service Provider, it is essential that the Service Provider should be trusted.

The license uses a standard language (the REL) to describe the conditions that have to be to meet to access a resource. However, the license does not provide the actual protection. This comes from encryption. The license creation process includes a step that generates an encryption/decryption (i.e. a symmetric) key for the resource. This key is then used to encrypt the resource. The decryption key is included in the license. Since the license is often included in the VDI whose resource it protects, the portion containing the decryption key must be encrypted as well. This encryption is performed using the AuthorizeUser Service Provider's public key. The encryption of the resource with the key included in the license, and the encryption of the specific part of the license is performed on the application/user side.

Figure 2 illustrates this process. In the first step, the application creates a license with the access rules governing the resource, either by directly accessing the middleware or by using a licensing tool that accesses the middleware on the application's behalf. The license that is returned in the second step contains a symmetric key. In step 3 the application uses this key to encrypt the resource. In step 4, the encrypted resource is stored in a repository where it can be accessed via a URI included in the VDI. The application then encrypts the license using the trusted AuthorizeUser SP public key. Finally, in step 6 the application creates the content VDI containing the encrypted license and a reference (URI) to the encrypted resource.

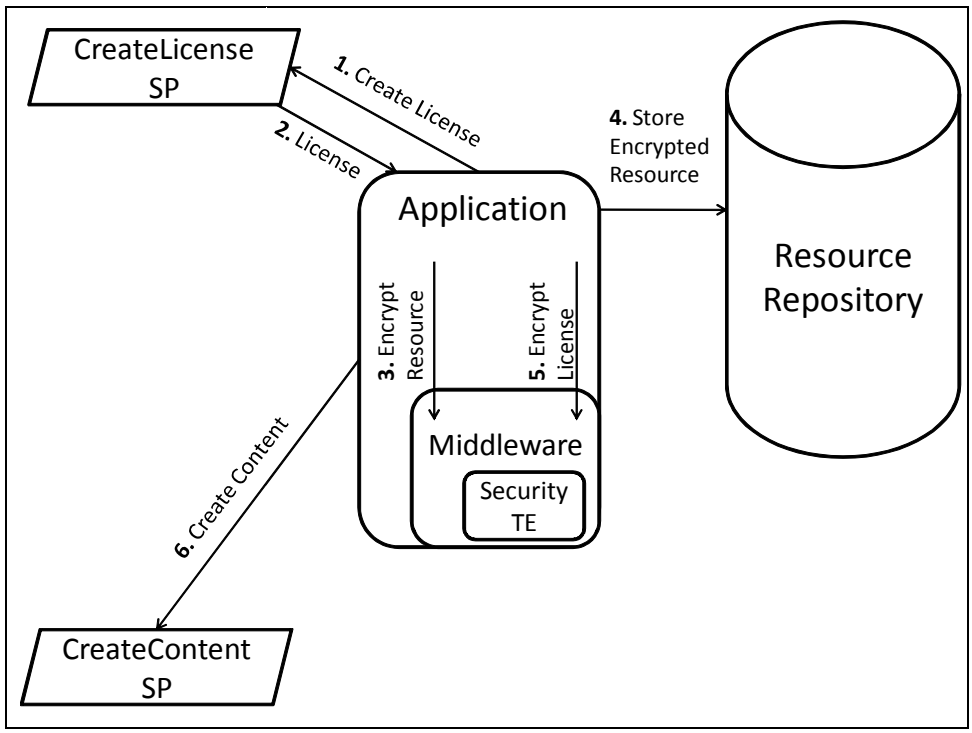


***Figure 2: Resource Protection with License***

Enforcement of the access control rules is performed by the AuthorizeUser SP. The protocol specifies that the entity requesting the authorization[3] passes to the SP the license protecting the content, the identifier or public key of the principal to be authorized, the right that the principal is attempting to exercise and the conditions under which the principal is attempting to exercise the right. As mentioned earlier, the license is encrypted with the public key of the AuthorizeUser SP. This SP now decrypts it and checks whether it contains a rule allowing the principal to exercise the requested right on the given resource. If this is the case, the decryption key is extracted from the license and passed to the requester in the protocol response.

Figure 3 depicts an example of an application accessing a protected resource. As a first step, the application fetches the VDI containing the resource reference from the network. It then opens the VDI, locates the license protecting the resource and requests authorization for the user who asked to access the resource. The user has previously been authenticated by the application which therefore knows who is trying to access the resource. If the license allows the user to exercise the requested right, the AuthorizeUser SP returns the symmetric key contained in the license. Finally, the application fetches the resource from the repository, using the URI included in the VDI, decrypts it with the symmetric key and delivers it to the requesting user.

[3]This is not necessarily the user –it may also be an application that needs to check whether a particular user is allowed to access a resource.

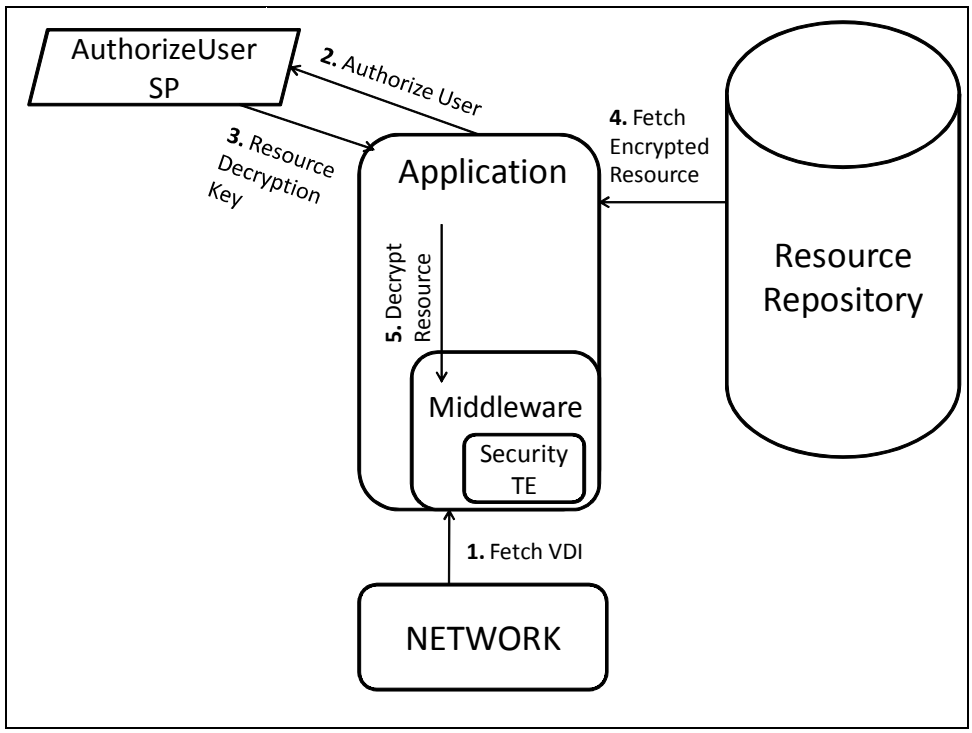


***Figure 3: Single User Authorization for Resource Access***

The previous example covers the case where the license grants a set of rights to a single user. However, a license may also grant rights to a user group, in which case the requesting user must prove that she is part of the group before obtaining authorization. In this case, the authorization is granted not to the user but to the group principal. Group membership is proved by means of licenses held by each member and signed by a trusted authority.

Group member authorization involves a two step workflow. The first step uses the CheckWithLicense Elementary Service to test whether the user is part of the group; the second uses the AuthorizeUser Elementary Service to check that the group has the requested right on the resource. The combination of the two services creates an aggregated service whose input is the request to the CheckWithLicense service and the license protecting the resource. The response is the response of the AuthorizeUser protocol. Given that a malicious user could try to get authorization by pretending to be a so-called *trusted root* (an authority which is trusted to verify that a particular user is a member of a group), the AuthorizeUser SP has to be sure that the request to the protocol request is not malicious. To achieve this, it requires that the request should be signed by a trusted aggregated Service Provider. At this point, the SP follows the same procedure it would follow to authorize an individual user.

The workflow is depicted in Figure 4. In the case shown here, the application requests an aggregated Service Provider to check that a requesting user is part of a group and, then, obtains authorization for the group. In step 2, the application provides the license proving that the user is part of the group, together with the license protecting the resource. The license providing group membership is used by the CheckWithLicense SP; the license protecting the

resource is used by the AuthorizeUser SP. If the authorization is successful, the decryption key is returned to the application, which fetches the resource and decrypts it.

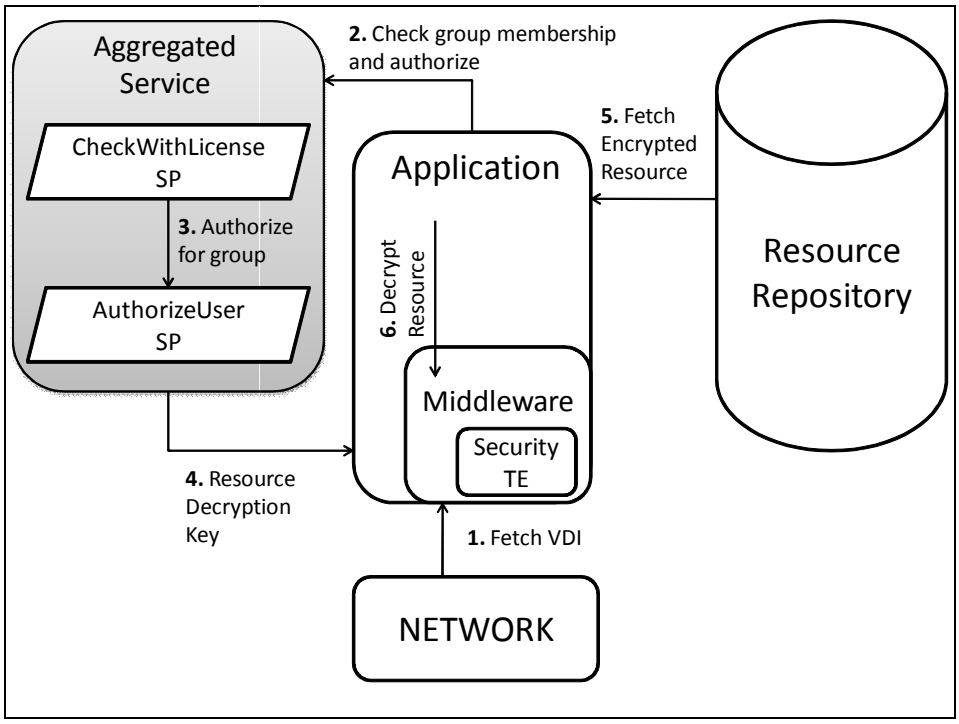


*Figure 4: Group Member Authorization*

## 3.4 Protocol and Technology Engines APIs

CONVERGENCE relies on MPEG-M for its middleware protocols. Where necessary, CONVERGENCE has proposed new protocols (Identify User, Post Content, Revoke Content) for inclusion in MPEG-M Part 4 [18], which is currently in the final stage of standardization. Specifications of the APIs for the corresponding Protocol Engines have been included in MPEG-M Part 2 [21].

In defining the protocols for the protocol and technology engines, CONVERGENCE has followed a unified approach. Each engine is split into a schema handler (mandatory) and a technology handler (optional). The schema handler is required for performing essential operations on the schema. For example, the Create Content Protocol Engine relies on the schema specified in MPEG-M Part 4 for the Create Content Elementary Service. Similarly, the REL technology engine relies on the REL schema, as specified in MPEG-21 Part 5. The technology handler provides the API for the operations that a given engine can perform. Thus, a Protocol Engine technology handler provides the API for making a call (local or remote) to the corresponding Elementary Service. For example, the technology handler for the Security technology engine provides functionality for signing, encrypting, decrypting etc.

### 3.4.1 CONVERGENCE Engines

CONVERGENCE makes use of the following engines.

#### 3.4.1.1 Protocol Engines

##### 3.4.1.1.1 BasePE

| **BasePE** | | **API: MPEG-M Part 2 [21]** |
|---|---|---|
| Engine dependencies | DigitalItemTE | Create and parse digital items |
| | MetadataTE | Create and parse MPEG-7 schema |
| | RELTE | Create and parse REL license expressions |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content. |

##### 3.4.1.1.2 AuthenticateContentPE

| **AuthenticateContentPE** | | **API: MPEG-M Part 2 [21]** |
|---|---|---|
| Relevant orchestrations | AuthenticateContentOrchestrator | |
| Engine dependencies | BasePE | AuthenticateContentPE extends BasePE |
| | OrchestratorTE | Provides the authenticate content orchestrator |

##### 3.4.1.1.3 AuthenticateUserPE

| **AuthenticateUserPE** | | **API: MPEG-M Part 2 [21]** |
|---|---|---|
| Relevant orchestrations | AuthenticateUserOrchestrator | |
| Engine dependencies | BasePE | AuthenticateUserPE extends BasePE |
| | OrchestratorTE | Provides the authenticate user orchestrator |

##### 3.4.1.1.4 AuthorizeUserPE

| **AuthorizeUserPE** | | **API: MPEG-M Part 2 [21]** |
|---|---|---|
| Relevant orchestrations | AuthorizeUserOrchestrator | |
| Engine dependencies | BasePE | AuthorizeUserPE extends BasePE |
| | OrchestratorTE | Provides the authorize user orchestrator |

#### 3.4.1.1.5 CheckWithLicensePE

| CheckWithLicensePE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | CheckWithLicenseOrchestrator | |
| Engine dependencies | BasePE | CheckWithLicensePE extends BasePE |
| | OrchestratorTE | Provides the check with license orchestrator |

#### 3.4.1.1.6 CreateContentPE

| CreateContentPE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | CreateContentOrchestrator | |
| Engine dependencies | BasePE | CreateContentPE extends BasePE |
| | OrchestratorTE | Provides the create content orchestrator |
| | IPMPTE | Create and parse Intellectual Property Management and Protection schema |
| | EventReportTE | Create and parse event report requests |

#### 3.4.1.1.7 CreateLicensePE

| CreateLicensePE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | CreateLicenseOrchestrator | |
| Engine dependencies | BasePE | CreateLicensePE extends BasePE |
| | OrchestratorTE | Provides the create license orchestrator |

#### 3.4.1.1.8 DescribeContentPE

| DescribeContentPE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | GetDescribeContentOrchestrator | |
| | SetDescribeContentOrchestrator | |
| Engine dependencies | BasePE | DescribeContentPE extends BasePE |
| | OrchestratorTE | Provides the get and set describe content orchestrators |

#### 3.4.1.1.9 IdentifyContentPE

| IdentifyContentPE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | IdentifyContentOrchestrator | |
| Engine dependencies | BasePE | IdentfyContentPE extends BasePE |
| | OrchestratorTE | Provides the identify content orchestrator |

#### 3.4.1.1.10 IdentifyDevicePE

| IdentifyDevicePE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | IdentifyDeviceOrchestrator | |
| Engine dependencies | BasePE | IdentifyDevicePE extends BasePE |
| | OrchestratorTE | Provides the identify device orchestrator |

#### 3.4.1.1.11 IdentifyUserPE

| IdentifyUserPE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | IdentifyUserOrchestrator | |
| Engine dependencies | BasePE | IdentifyUserPE extends BasePE |
| | OrchestratorTE | Provides the identify user orchestrator |

#### 3.4.1.1.12 PackageContentPE

| PackageContentPE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | PackageContentOrchestrator | |
| Engine dependencies | BasePE | PackageContentPE extends BasePE |
| | OrchestratorTE | Provides the package content orchestrator |
| | MP21FFTE | Create and parse MPEG-21 formatted files |

#### 3.4.1.1.13 PostContentPE

| PostContentPE | API: MPEG-M Part 2 [21] |
|---|---|
| Relevant orchestrations | PostContentOrchestrator |

| | | |
|---|---|---|
| Engine dependencies | BasePE | PostContentPE extends BasePE |
| | OrchestratorTE | Provides the post content orchestrator |

#### 3.4.1.1.14 RequestContentPE

| **RequestContentPE** | | **API: MPEG-M Part 2 [21]** |
|---|---|---|
| Relevant orchestrations | RequestContentOrchestrator | |
| Engine dependencies | BasePE | RequestContentPE extends BasePE |
| | OrchestratorTE | Provides the request content orchestrator |

#### 3.4.1.1.15 RequestEventPE

| **RequestEventPE** | | **API: MPEG-M Part 2 [21]** |
|---|---|---|
| Relevant orchestrations | RequestEventOrchestrator | |
| Engine dependencies | BasePE | RequestEventPE extends BasePE |
| | OrchestratorTE | Provides the request event orchestrator |
| | EventReportTE | Create and parse event reports and event repost requests |

#### 3.4.1.1.16 RevokeContentPE

| **RevokeContentPE** | | **API: MPEG-M Part 2 [21]** |
|---|---|---|
| Relevant orchestrations | RevokeContentOrchestrator | |
| Engine dependencies | BasePE | RevokeContentPE extends BasePE |
| | OrchestratorTE | Provides the revoke content orchestrator |

#### 3.4.1.1.17 SearchContentPE

| **SearchContentPE** | | **API: MPEG-M Part 2 [21]** |
|---|---|---|
| Relevant orchestrations | SearchContentOrchestrator | |
| Engine dependencies | BasePE | SearchContentPE extends BasePE |
| | OrchestratorTE | Provides the search content orchestrator |
| | SearchTE | Create and parse MPEG Formatted Queries |

#### 3.4.1.1.18 StroreContentPE

| StoreContentPE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | StoreContentOrchestrator | |
| Engine dependencies | BasePE | StoreContentPE extends BasePE |
| | OrchestratorTE | Provides the store content orchestrator |

#### 3.4.1.1.19 StoreEventPE

| StoreEventPE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Relevant orchestrations | StoreEventOrchestrator | |
| Engine dependencies | BasePE | StoreEventPE extends BasePE |
| | OrchestratorTE | OrchestratorTE provides the store event orchestrator |
| | EventReportTE | Create and parse event reports and event repost requests |

### 3.4.1.2 Technology Engines

#### 3.4.1.2.1 CDSTE

| CDSTE | | API: see 3.4.2.1 |
|---|---|---|
| Engine dependencies | - | |

#### 3.4.1.2.2 ConetTE

| ConetTE | | API: see 3.4.2.2 |
|---|---|---|
| Engine dependencies | - | |

#### 3.4.1.2.3 ConvergenceMetadataTE

| ConvergenceMetadataTE | | API: see 3.4.2.3 |
|---|---|---|
| Engine dependencies | DigitalItemTE | Create and parse structured data |
| | SearchTE | Create and parse MPEG formatted input queries |

#### 3.4.1.2.4 DigitalItemTE

| DigitalItemTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Engine dependencies | EventReportTE | Create and parse event report schemas |
| | MetadataTE | Create and parse MPEG-7 metadata |
| | RELTE | Create and parse REL license expressions |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content |
| | IPMP TE | Create and parse Intellectual Property Management and Protection schema |

#### 3.4.1.2.5 EventReportTE

| EventReportTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Engine dependencies | RELTE | Create and parse REL license expressions |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content |

#### 3.4.1.2.6 IPMPTE

| IPMPTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Engine dependencies | RELTE | Create and parse REL license expressions |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content |

#### 3.4.1.2.7 MatchTE

| MatchTE | API: see 3.4.2.5 |
|---|---|
| Engine dependencies | - |

#### 3.4.1.2.8 MediaFrameworkTE

| MediaFrameworkTE | API: MPEG-M Part 2 [21] |
|---|---|
| Engine dependencies | - |

#### 3.4.1.2.9 MetadataTE

| MetadataTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Engine dependencies | - | |

#### 3.4.1.2.10 MP21FFTE

| MP21FFTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Engine dependencies | DigitalItemTE | Create and parse digital items |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content |

#### 3.4.1.2.11 MpqfTE

| MpqfTE | | API: see 3.4.2.6 |
|---|---|---|
| Engine dependencies | SearchTE | MpqfTE extends SearchTE |

#### 3.4.1.2.12 OverlayTE

| OverlayTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Engine dependencies | DigitalItemTE | Create and parse digital items |
| | ConetTE | Send and receive network messages |

#### 3.4.1.2.13 RELTE

| RELTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Engine dependencies | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content |

#### 3.4.1.2.14 SearchTE

| SearchTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Engine dependencies | - | |

#### 3.4.1.2.15 SecurityTE

| SecurityTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Engine dependencies | - | |

#### 3.4.1.2.16 VDITE

| VDITE | | API: see 3.4.2.7 |
|---|---|---|
| Engine dependencies | DigitalItemTE | VDITE extends DigitalItemTE |

### 3.4.1.3 Orchestrator Engines

#### 3.4.1.3.1 OrchestratorTE

| OrchestratorTE | | API: MPEG-M Part 2 [21] |
|---|---|---|
| Orchestrations | Engine dependencies | |
| AuthenticateContentOrchestrator | DigitalItemTE | Create and parse digital items |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content. |
| AuthenticateUserOrchestrator | MetadataTE | Create and parse MPEG-7 schema |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content. |
| AuthorizeUserOrchestrator | RELTE | Create and parse REL license expressions |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content. |
| CheckWithLicenseOrchestrator | RELTE | Create and parse REL license expressions |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content. |
| CreateContentOrchestrator | MetadataTE | Create and parse MPEG-7 schema |
| | DigitalItemTE | Create and parse digital items |
| | RELTE | Create and parse REL license expressions |
| | IPMP TE | Create and parse Intellectual Property Management and Protection schema |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and |

| | | |
|---|---|---|
| | | encrypted content. |
| CreateLicenseOrchestrator | RELTE | Create and parse REL license expressions |
| GetDescribeContentOrchestrator | MetadataTE | Create and parse MPEG-7 schema |
| SetDescribeContentOrchestrator | MetadataTE | Create and parse MPEG-7 schema |
| IdentifyContentOrchestrator | DigitalItemTE | Create and parse digital items |
| IdentifyDeviceOrchestrator | IPMP TE | Create and parse Intellectual Property Management and Protection schema |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content. |
| IdentifyUserOrchestrator | MetadataTE | Create and parse MPEG-7 schema |
| | SecurityTE | Create and parse XML signatures, certificates, SAML assertions and encrypted content. |
| PackageContentOrchestrator | DigitalItemTE | Create and parse digital items |
| | MP21FFTE | Create and parse MPEG-21 formatted files |
| PostContentOrchestrator | DigitalItemTE | Create and parse digital items |
| | OverlayTE | Post to overlay network |
| RequestContentOrchestrator | DigitalItemTE | Create and parse digital items |
| RequestEventOrchestrator | EventReportTE | Create and parse event report schemas |
| RevokeContentOrchestrator | DigitalItemTE | Create and parse digital items |
| SearchContentOrchestrator | MetadataTE | Create and parse MPEG-7 schema |
| | SearchTE | Create and parse MPEG formatted queries |
| StoreContentOrchestrator | DigitalItemTE | Create and parse digital items |
| StoreEventOrchestrator | EventReportTE | Create and parse event report schemas |

#### *3.4.1.3.2 ConvergenceOrchestratorTE*

| **ConvergenceOrchestratorTE** | | **API: see 3.4.2.4** |
|---|---|---|
| Engine dependencies | OrchestratorTE | ConvergenceOrchestratorTE extends OrchestratorTE |
| Orchestrations | Engine dependencies | |

| | | |
|---|---|---|
| ConetOrchestrator | ConetTE | Advertise and fetch resources in CONET |
| EventReportOrchestrator | EventReportTE | Generate event reports |
| MatchOrchestrator | MatchTE | Store and match incoming subscriptions and publications |
| | VDITE | Parse P/S-VDIs |
| | ConvergenceMetadataTE | Parse Convergence Metadata |
| | CDSTE | Expand P/S-VDI metadata |
| | EventReportTE | Store P/S-VDI event report requests |
| | MpqfTE | Parse MPEG formatted subscription queries |
| MetadataOrchestrator | ConvergenceMetadataTE | Create Convergence Metadata |
| | MpqfTE | Create MPEG formatted subscription queries |
| OverlayOrchestrator | VDITE | Parse P/S-VDIs |
| | OverlayTE | Propagate P/S-VDIs in the semantic overlay |
| TimerOrchestrator | - | |
| VdiOrchestrator | VDITE | Parse P/S-VDIs |
| | IdentifyContentPE | Identify VDIs |
| | CreateContentPE | Create VDIs |

### *3.4.2 CONVERGENCE Engines APIs*

#### 3.4.2.1 CDSTE

The Community Dictionary Service Technology Engine (CDS TE) provides semantic services to the CONVERGENCE middleware. There are basically two types of services: management services for ontologies and exploration services for the CDS's knowledge base.

The CDSTE provides the following interfaces:

- *Entity Explorer*: an interface defining methods to search for ontology entities stored in the CDS repository
- *Semantic Expander*: an interface that defines methods for expanding semantic metadata exploiting the CDS knowledge base
- *Knowledge Manager:* an interface defining methods to maintain the CDS knowledge base

- *Repository Manager*: an interface that defines methods for storing and accessing meta-models
- *Inference Manager:* an interface that defines methods for reasoning over the CDS knowledge base and on demand inference over metadata
- *CommunicationManager:* an interface defining methods to access the CDS Engine remotely.

#### 3.4.2.2 CoNetTE

The CoNet Technology Engine (CoNet TE) provides the CONVERGENCE Middleware with access to the features of the CoNet, based on Information Centric Networking. Specifically, it provides support for:

- Storing a resource in the CoNet with a given Network Identifier (NID)
- Revoking a content with a given NID from the CoNet
- Retrieving a resource corresponding to the input NID from the CoNet
- Advertising a Service Access Point(SAP) for receiving unNamed-data, providing the callBack to be invoked when unNamed-data are received
- Sending unNamed-data towards a NID (the unNamed-data are expressed as a byte array)
- Sending unNamed-data towards a Location ID (LID)

#### 3.4.2.3 ConvergenceMetadataTE

The ConvergenceMetadataEngine defines methods for creating and parsing the CONVERGENCE metadata schema and provides the following interfaces:

- *ConvergenceMetadataParser*: an interface that defines methods for parsing convergence metadata
- *RdfMetadataParser*: an interface that defines methods for parsing RDF metadata
- *ConvergenceMetadata*: an interface that defines methods for creating convergence metadata structures
- ResourceMetadata: an interface defining methods for creating and parsing R-VDI metadata
- *PublicationMetadata*: an interface defining methods for creating and parsing P-VDI metadata
- *SubscriptionMetadata*: an interface defining methods for creating and parsing S-VDI metadata

#### 3.4.2.4 ConvergenceOrchestratorTE

The ConvergenceOrchestratorTE extends the OrchestratorTE and provides orchestrations for specific CONVERGENCE operations. The ConvergenceOrchestratorTE provides the following orchestrations:

- *ConetOrchestrator*: methods for advertising and fetching resources from CoNet
- *EventReportOrchestrator*: methods for generating event reports
- *MatchOrchestrator*: methods for parsing and matching S-VDIs with P-VDIs
- *MetadataOrchestrator*: methods for creating R/P/S-VI metadata
- *OverlayOrchestrator*: methods for handling and propagating P/S-VDIs in the semantic overlay
- *TimerOrchestrator*: methods for scheduling and running periodic middleware tasks
- *VdiOrchestrator*: methods for creating, identifying and storing VDIs

#### 3.4.2.5 MatchTE

The Match Technology Engine (TE) is responsible for performing matches between subscriptions and publications. The Match TE offers the following interfaces:

- *Publication Manager*: an interface that defines methods for indexing publications
- *Subscription Manager*: an interface that defines methods for indexing subscriptions
- *Rdf Repository Manager*: an interface that defines methods for storing and matching rdf metadata of publications with sparql subscription queries
- *Field Value Repository Manager*: an interface that defines methods for storing and matching field/value publication metadata with field/value subscription queries

#### 3.4.2.6 MpqfTE

The Mpqf Engine interface defines methods for creating and parsing queries expressed in the MPEG Query Format. The Mpqf Engine provides the following interfaces:

- *FieldValueQueryEditor*: an interface that defines methods for creating and parsing field/value queries
- *SparqlQueryEditor:* an interface defining methods for creating and parsing SPARQL queries.

#### 3.4.2.7 VDITE

The VDIEngine extends the DigitalItem engine and provides methods for creating and parsing inter-VDI relationship descriptors. The VDIEngine provides the following interfaces:

- *Relationship*: an interface that defines methods for creating and parsing inter-VDI relationships.

# 4 Network Support for Video Streaming

## 4.1 Video Streaming Services over CONET

In this section, we describe the CONVERGENCE support for video streaming services over the CONET. With respect to the solution described in D5.1 the solution described here represents a major improvement. These services provide an example of exploitation of CONET protocols and APIs and are included in the prototype and in the demo shown in relevant public events.

As shown in Figure 5, CONVERGENCE uses video streams structured in "segments"; where each segment is a piece of the whole video (e.g. 3 sec), has a unique Network Identifier (NID), i.e. "foo.eu/video001.ts", and is contained in a single named-data CIU.

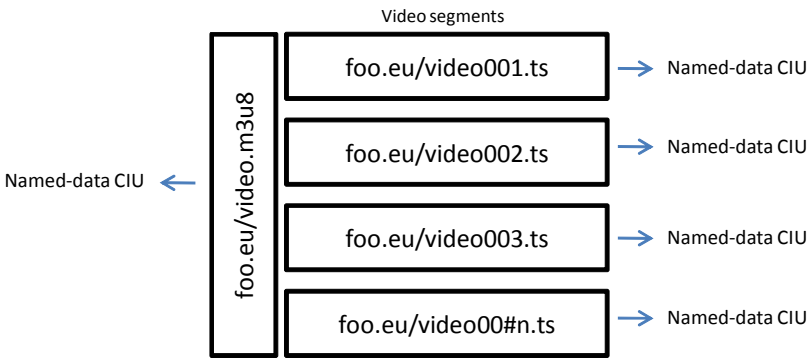


Figure 5: Segmented video stream

This is the same streaming scheme used by the Apple HTTP Live streaming protocol [1] and by the forthcoming MPEG DASH [2]; in both cases, the unique content-name of a video-segment is a plain URL that we can map to a NID by removing the protocol part, i.e. HTTP.

Apple HTTP Live streaming is currently supported by Apple devices, the latest Android phones, and VLC 2.0 applications for PC. MPEG DASH is at an early stage of prototyping.

In CONVERGENCE, the named-data CIUs of the video are stored in a CONET serving node, the CONET routing plane then configures the routing tables of the border-nodes so as to forward requests for the segments towards the serving node.

Video streaming can be secured via a smart card. In this scheme, each video segment is encrypted with a symmetric key, which is "wrapped" with the designated recipients' public keys. This means that only authorized users can see the stream. When a video segment is downloaded, the user's smart card decrypts it, by using the symmetric key extracted from the wrapped key.

### 4.1.1 Individual Streaming

To view the video, a client (e.g., VLC on an end-node) begins by downloading a "m3u8" playlist file (see Figure 6). The m3u8 file contains the NIDs of the video segments composing the stream together with TAGs, which support the timing of the streaming operation.

After downloading the playlist, the client starts to continuously "pre-fetch" a small set of video segments, always remaining slightly in advance of the video that is playing. The client allows the user to pause or jump forward/backward in the sequence of video segments.

In case of a popular video, CONET'S network-caching functionality stores many named-data items in nodes close to the user, dramatically lowering network traffic and the load on the server.

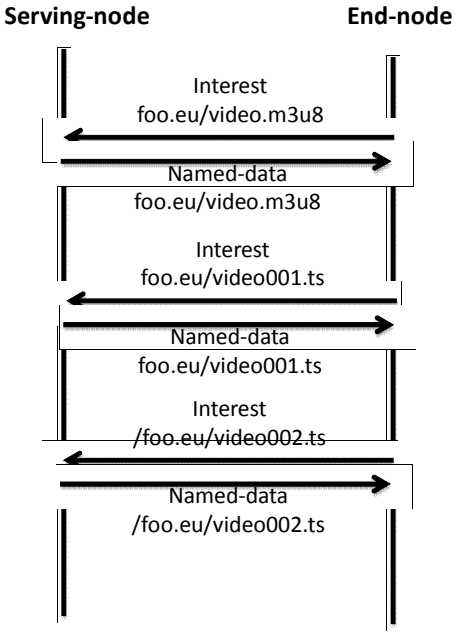


*Figure 6: Individual Streaming*

### 4.1.2 Cooperative Streaming for Cellular Networks

In this section, we propose a cooperative video streaming solution for mobile cellular devices, which can be used to offload the radio cellular interface. As we will see, the scheme we propose is based on content-centric concepts. Offloading is achieved by exploiting the routing-by-name and in-network caching functionality of an Information Centric Network (ICN) such as our CONET.

In the scenario we consider here, a group of neighbouring users use their mobile devices to view the same video, at the same time. The video is offered by a serving-node located in the fixed network.

The mobile devices are CONET end-nodes, connected to the fixed network by means of a cellular radio interface and connected to each other by means of a local wireless technology, e.g. an *ad hoc* Wi-Fi connection.

Thanks to our CONET application, each video segment is downloaded by a single user, who then shares it over the ad-hoc Wi-Fi network. Compared to traditional HTTP streaming or CONET individual streaming (see section 4.1.1), this has the effect of offloading the cellular radio interface.

The basic underlying idea is that for each segment, one device plays the role of local-proxy for other neighbouring devices.

All requests for a given segment are routed-by-name towards the local-proxy device, whose content-cache becomes the common en-route cache for the entire group. The first request is forwarded to and served by the origin streaming server, through the cellular interface. All subsequent requests are served by the content cache via Wi-Fi.

To share the cost of the cellular connection in a fair way, each device belonging to a member of the group plays the role of local-proxy for different sets of video segments, equalizing the number of segments downloaded over the cellular interface by each device.

Currently, we do not consider the presence of free-riders or misbehaving users. However, we believe that this problem could be handled by filtering incoming content requests, e.g. by adapting P2P strategies [4].

Figure 7 illustrates our ICN video Streaming Application (ICN-SA) for the scenario described above.

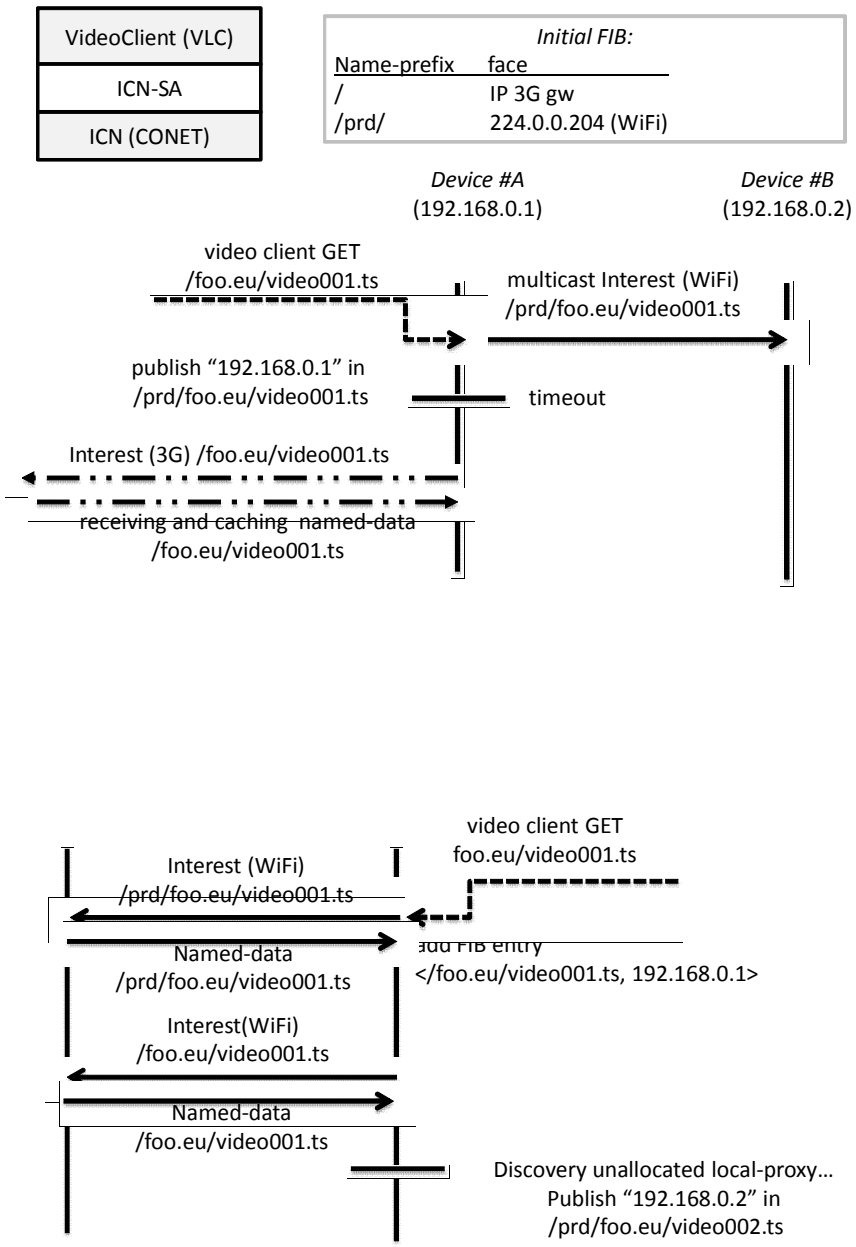


***Figure 7- Cooperative video streaming for cellular networks***

The streaming server, not shown in the figure, is located in the fixed network. This server acts as a repository that publishes all of a video's segments/name-data, including its m3u8 file. The segments have the unique network identifier (NID) "/foo.eu/video00x.ts", where x is the segment number.

Consider the case of two neighbouring 3G mobile devices, named #A and #B. Both devices have a preloaded CONET FIB containing the default "/" entry towards the CONET gateway provided by the 3G network (3G-gateway). Both also contain the "/prd/" entry used for proximity-route-discovery (prd) operation, over the Wi-Fi multicast channel 224.0.0.204.

Assume that both devices have already downloaded the m3u8 file and that user #A plays the video slightly in advance with respect to user #B.

Video client #A (e.g., VLC) requests the first segment: /foo.eu/video001.ts. The request is handled by ICN-SA #A, which starts a proximity-route-discovery procedure aimed at discovering if another device has been set as local-proxy of /foo.eu/video001.ts.

CONET mechanisms can also be used to support the proximity-route-discovery procedure. When a device sets itself as a local-proxy of the segment /foo.eu/video00x.ts, it publishes a proximity-route-content, named with the same NID of the content to be routed (/foo.eu/video00x.ts), but with the special prefix "/prd/", i.e. /prd/foo.eu/video00x.ts. The payload of the proximity-route-content contains information on the interface that the local-proxy device is offering, e.g., its IP address 192.168.0.1.

To discover a proximity-route for /foo.eu/video001.ts, ICN-SA #A requests the related proximity-route-content /prd/foo.eu/video001.ts on the Wi-Fi multicast interface. Since there is no other local-proxy for /foo.eu/video001.ts, the proximity-route-discovery fails after a timeout. Consequently, ICN-SA #A configures the device #A as a local-proxy for /foo.eu/video001.ts, by publishing the related proximity-route-content. ICN-SA #A then requests the content /foo.eu/video001.ts from CONET, which routes-by-name the request, using the default FIB entry towards the 3G-gateway. The content request is received by the video server, cached by device #A and passed to video client #A.

After a few seconds, video client #B also requests /foo.eu/video001.ts. ICN-SA #B performs the proximity-route-discovery procedure and obtains the proximity-route-content /prd/foo.eu/video001.ts. Using the payload of this content, the ICN-SA #B adds the route </foo.eu/video001.ts, 192.168.0.1> to its CONET FIB. Then, ICN-SA #B requests the content /foo.eu/video001.ts from the CONET. Thanks to its longest prefix matching policy, CONET routes the request to Wi-Fi 192.168.0.1 (device #A), rather than to the 3G-gateway. The content cache of the device #A sends back the content to ICN-SA #B via Wi-Fi.

As soon as it receives a segment, ICN-SA #B offers itself as a possible local-proxy for other segments, ensuring that the cost of the cellular connection is divided fairly. To this end, it searches proximity-routes not yet offered by other devices; in our figure, it discovers that the proximity-route-content for segment /foo.eu/video002.ts is not available, and sets itself as a local-proxy for the segment. To do this, it publishes the related proximity-route-content ("/prd/foo.eu/video002.ts"). It then repeats the same procedure for all the other video segments.

# 5 Bibliography


[1] R. Pantos, “HTTP Live Streaming” Internet draft, http://tools.ietf.org/html/draft-pantos-http-live-streaming-01

[2] “Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats”, ISO/IEC 23009-1:2012

[3] R. Pantos, “HTTP Live Streaming” Internet draft, http://tools.ietf.org/html/draft-pantos-http-live-streaming-01

[4] J. Mol, J. Pouwelse, M. Meulpolder, D. Epema, and H. Sips, “Give-to-get: An algorithm for P2P video-on-demand," ACM MMCN '08

[5] 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.

[6] 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.

[7] 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.

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

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

[10] 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.

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

[12] 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.

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

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

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

[16] 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.

[17] 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.

[18] ISO/IEC FDIS 23006-4:2012(E), Information technology – Multimedia service platform technologies – Part 4: Elementary Services.

[19] ISO/IEC 21000-6:2004, Information technology - Multimedia framework (MPEG-21) - Part 6: Rights Data Dictionary.

[20] OASIS, Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005.

[21] ISO/IEC CD 23006-2, Information Technology – Multimedia service platform technologies – Part 2: MPEG extensible middleware (MXM) API.