Architectural Styles of Content Negotiation in The Profile Dimension

No Author Given

No Institute Given

Abstract. Identification, interaction and data formats are the pillars of the Web architecture. Content negotiation plays a major role in the interaction part by providing the means by which a server responds to a client request for a resource with the appropriate representation that satisfies a set of constraints. Content negotiation by media type and language is established on the Web. The under development profile dimension for content negotiation describes structural and/or semantic constraints on resource representations. In the Semantic Web, a client may be interested in negotiating RDF graphs that conform to a profile (e.g. represented as a SHACL shape graph). In the request-response model of HTTP, different architectural styles are possible to perform content negotiation, each with its advantages and disadvantages. In this paper, we review existing contributions related to content negotiation by profile, with a particular focus on the ongoing efforts led by the Dataset Exchange Working Group and the Internet Engineering Task Force. We then propose an overview of the possible architectural styles for content negotiation by profile, and discuss trade-offs for using one over the other. Using an incremental use case and scenarios, we show how some architectural styles would address several requirements that have not yet been addressed. An implementation and experiments are proposed to demonstrate the applicability of some of the architectural styles.

Keywords: Content negotiation· Negotiation by profile· RDF· Semantic Validation· SHACL· HTTP· Architectural Style

1 Introduction

Content negotiation (CN) plays a central role in the Web architecture. Since the inception of the World Wide Web (WWW), CN has been part of it, as shown by the first publication about WWW [7]. CN happens when a client requests a representation of a resource (e.g. issuing an HTTP GET) accompanied by constraints that the returned representation should satisfy [17, Section 12]. Sometimes, a server has several representations for the same resource, so a negotiation phase happens: in some cases, the server decides which representation is the most appropriate given the constraints and returns it directly; sometimes, the server provides a redirection to another resource or representation that could satisfy the client; sometimes the server simply informs the client of all the available representations, possibly with weights indicating a preference.

While initially, CN was made to allow a client to request a resource in a specified format, we can now view CN in a more general way. Even with the same media type, several representations of the same resource may exist to suit different constraints. For instance, multimedia documents may exist with different encoding qualities, structured data may adhere to different schemas, or formal knowledge may be equivalently represented with different axioms. In this paper, we are interested in content negotiation by *profile*, as explored by the W3C Data Exchange working group (DXWG). A profile denotes a *description of structural and/or semantic constraints on resource representations that apply in addition to the constraints inherently indicated by their media type* [44]. More precisely, we examine how CN by profile can be operated in different *architectural styles* (AS), with our investigation encompassing what the DXWG is standardising as well as more advanced forms of CN.

To motivate the need for more advanced CN, we consider the case of a large, multinational building company that aggregates data about its assets coming from many sources. Each building is described in several formats, but also using different building ontologies so that it can conform to different local standards. We will call this use case *UC-A* and we will use it to illustrate scenarios relying on different architectural styles. Additionally, we consider use case *UC-B* where a mediating server provides different RDF graphs for the same URI. The graphs are representations of the same resource, but they are hosted on different servers. The mediating server is able to identify distributed representations of the same resource by relying on `sameAs` links.

Our contributions match the organisation of the paper. In Section 2, we present past and current work on how to convey profile preferences. In Section 3 we review AS options that can be considered for implementing CN by profile in practice, while highlighting the advantages and drawbacks of each solution. The analysis synthesises how existing literature and implementations relate to the relevant AS with Section 4 dedicated to the implementations of the DXWG negotiation by profile specification. In Section 5, we show how UC-B can take advantage of a constraint definition language (viz. SHACL) to implement CN by profile. Our implementation of this scenario demonstrates a unique combination of AS that has not been proposed before.

2 State of the art on conveying profile information

Content negotiation on the Web is initiated by requesting a resource from its URI and additionally specifying preferences about the desired representation of that resource. The DXWG describes several use cases that require profile-based content negotiation [37, see e.g. use case ID2 and ID3]. However, for the Semantic Web community, the use case of requesting an RDF representation with some expected vocabularies is older. In 2006 already, a thread on the Semantic Web mailing list discussed the topic.[1] Two essential components are important to

[1] Semantic Web mailing list archives: *expectations of vocabulary* `https://lists.w3.org/Archives/Public/semantic-web/2006Jul/0062.html`

convey the preferences of the client: a way of representing the preferences (in our case, the definition of a profile) and a way of transmitting them. We examine the state of the art for the two components.

2.1 Specifying a profile

HTTP currently defines CN in four *dimensions*: *media type*, *language*, *encoding*, and *character set* [17, Section 12.5]. Among these, media types can serve as a way to specify a profile. Media type definitions can add certain structural constraints to a more generic format, hence creating a "profile" for it. For instance, in its XML Media Types proposal [33], the IETF network working group introduced the use of the convention of adding (`+xml`) for newly created media types that are XML-based, which allows for generic XML processing. This suffix convention has since become the de facto practice for other structuring syntax (e.g. `+json`, `+zip`, `+jwt`) [18, Section 4.2.8]. Even if we ignore the constraints of the required registration process, it is still impractical to define a new media type for all possible profiles, especially with a broad sense of "profile" where constraints could be customised per application. Moreover, a profile would then be tied to the media type, whereas the two dimensions should be orthogonal, which is why we can observe that some media types have the same prefix but differ only in their suffixes (e.g. `senml+xml`, `senml+exi`, `senml+json`, `senml+cbor`). In this case, if a client wants to negotiate a profile disregarding its serialisation, it must specify all possible media types.

In principle, it could be possible to append vocabulary prefixes to the media-type token (e.g., `text/foaf+schema+org+turtle`) to form more fine-grained constraints while keeping the exact same architecture for CN. However, the limitation of such approach should be obvious and still this would not cover many more kinds of specific constraints that one may have, as highlighted by Verborgh [47]. This is also known as the media type explosion problem.[2]

Indicating profiles in the media type is one way of specifying a profile. The W3C Composite Capability/Preference Profiles (CC/PP) [20] is an RDF-based standard for describing device capabilities and user preferences. A server can use such information to customise the served content. The Friend of a Friend vocabulary (FOAF) have been proposed as a way to define a personal profile that can be exploited by servers to improve Web browsing experience [3]. The DXWG defines a vocabulary that can express metadata about profiles [5]. The DXWG also mentioned that a constraint or schema language (e.g. SHACL [21]) can be used to specify a profile [44]. The group also notes that a human-readable document (e.g. in HTML or PDF) may serve as the specification of a profile. In each of these cases, the profile can be referred to by serialising it appropriately in a payload, or by way of a URI.

While it is preferable to identify profiles by URIs that dereference to a description [42, Section 6.2][44, Section 4.1], it is possible to use URIs that do

[2] media type explosion problem (Last paragraph): https://lists.w3.org/Archives/Public/semantic-web/2006Jul/0097.html

not. However, means should be provided for clients and servers to understand their meaning. A server can advertise a repository containing the profiles it can handle, e.g. SpacePrez[3] or CC/PP repositories [20, Section A.1].

Some profile may define a set of constraints of varying importance. In a CN implementation using CC/PP and WAP UAProf [11], the author extends the Apache Web server by providing the ability to configure the constraints as a *hard* or *soft* constraint using a boolean `mustmatch` attribute. Similarly, in SHACL, the specification provides the means to define a *severity* for each shape [21, Section 2.1.4]. This offers ways of enabling more flexible negotiation, as we will see in Section 3.4. In [45], the authors proposed to rank different RDF graphs with respect to a SHACL schema with severities, which can serve in the negotiation mechanism.

2.2 Conveying profile preferences

A client expecting conformance to profiles must convey them to the server. We build on top of the DXWG group to classify existing approaches [42, Section 5]. Two main options are found in the state of the art: Header-based, that is when the profile is indicated in the value of a HTTP header; and URI-based approaches, where profile information is given in a portion of the URI [46].

Header-based. The DXWG identified several existing standards that rely on HTTP headers and that could be used to convey profile preferences, but argued that they are insufficient for fully capturing the needs of CN by profile. Media types can provide additional parameters [17, Section 12.5.1] that can be appended to the value of the `Accept` header, particularly the `profile` parameter (e.g. `application/ld+json;profile="http://www.w3.org/ns/json-ld#flattened"`).[4] However, not all media types allow such parameter and a profile of this kind is media-type-specific. The `Link` header [35] together with the `rel="profile"` relation type [49], as well as the `Prefer` header [39] could provide a profile preference, but neither can convey quality information (q-values) [17, Section 12.4.2]. Note, however, that the `Prefer` header also allows for processing preferences that have hard and soft constraints using `handling=strict` and `handling=lenient` [39, Section 4.4]. The `Accept-Profile` header [47] is now DXWG's preferred proposal [42, Section 7.2] for conveying preferences regarding resource representation profiles. A separate IETF document explains in more detail how to indicate, discover, negotiate and write profiled representations [44].

Other suggestions were made in less formal discussion forums related to CN. `Accept-Vocabulary` was proposed to convey accepted vocabulary in an

[3] Profiles recognised by SpacePrez: https://gnaf.linked.fsdf.org.au/profiles

[4] RFC 7284 [24] defines a IANA registry for profile URIs https://www.iana.org/assignments/profile-uris/profile-uris.xhtml

RDF representation[5]. `Accepts-Vocab`[6] is similar but for JSON representations. `Accept-schema` [41] would provide metadata schema negotiation. In the appendices of the XML media type specification, the suggestion "How about using a CN tag instead (e.g. accept-features: syntax=xml)?" could be generalised to convey profile indication.

In contrast to previous options that provide constraints on the expected content, Composite Capability/Preference Profiles (CC/PP) [20] is a standard for describing device capabilities and user preferences. A server can use such information to customise the served content. The `Profile` header can be used to refer to such profiles via its URI [36, Section 5.2]. Similarly, `XFOAF` [3] is a header proposal used to allow a client to communicate its FOAF profile (via its URI) to a server. `Accept-Lowering-Rule` [25] points to the URI of a lowering rule the server should use to serialise a RDF graph.

URI based. *Query string parameters* can be used to convey profile preferences in the URI [8, Section 3.4], e.g. `http://domain.com/path/to/res?profile=profile_info`. Here, when negotiating for the resource identified by `http://domain.com/path/to/res`, profile preference is conveyed using query parameter `profile`. Such use can be observed in several settings.[7,8,9] The DXWG proposes the `_profile` query parameter to convey profile preferences [42, Section 7.3].

Suffix pattern matching [46, Section 3.3] uses the part of the URI after a specific character to convey profile preferences, e.g. `http://domain.com/path/to/res.profile_info`. In this example we use '`.`' as a special character to indicate the start of the preferred profile. Other conventions could be used to signal the end of the resource identification and the start of the profile preferences. An example of such an approach is the Archival Resource Key Identifier Scheme [22], where qualifiers are added as a suffix to the URI. Istex is a French scientific archive that uses such an approach to serve various XML representations [48], including Text Encoding and Interchange [38].

Although these different contributions differ in their final objectives, the common point that unites them is the use of CN, as highlighted over the years by several working groups in their best practice documents [9,16,10], where each use case involves some CN AS. However, to the best of our knowledge, no previous work has attempted to compile the different CN AS. In particular, to illustrate how they can be used in the profile dimension, and how semantic validation can

[5] Vocabulary negotiation header for RDF: `https://lists.w3.org/Archives/Public/semantic-web/2006Jul/0068.html`

[6] Vocabulary negotiation header for JSON `https://code.sgo.to/2015/09/01/vocabulary-negotiation.html`

[7] Linked Data API using the `_view` query parameter: `https://github.com/UKGovLD/linked-data-api/blob/wiki/Specification.md`

[8] The Open Archives Initiative Protocol for Metadata Harvesting uses the `Verb` query parameter [23]

[9] The OGC Catalogue Service for the Web Standard uses the `outputSchema` query parameter [34]

be applied and integrated in different CN scenarios. This is what we present in the following section.

3 An Overview of CN Architectural Styles

In this section, we discuss possible AS for handling CN. In particular, we show how some of them can make use of the current web standards and infrastructure, and illustrate these based on UC-A and the current efforts of DXWG [42] and IETF [44]. The AS presented here, though illustrated via RDF and SHACL, are general enough to cover negotiation over any structured data model.

To better understand the different CN AS, we start by defining a general use case that will serve as the basis for the following subsections. We then consider several scenarios, each with specific requirements and needs. We then propose a solution for each scenario and illustrate the appropriate AS.

3.1 Illustrative scenario

The company All-About-Buildings is a Qatar-based multinational who uses URIs to identify buildings all over the world. For example, the building at address "55 Al Arab St, Doha, Qatar" is identified by `<http://All-About-Buildings.com/buildings/b55112>`. The company adopts three different ontologies in order to support the needs of different customers and describe the buildings accordingly (see Figure 1).

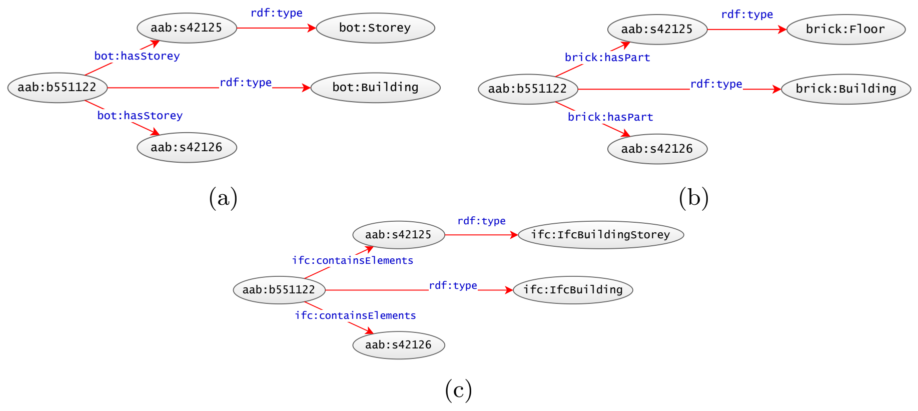

Fig. 1: RDF graphs describing a building according to 3 ontologies: (a) the Building Topology Ontology (BOT); (b) the Brick schema; (c) the ifcOWL ontology

Depending on the AS, there are different ways for the client to request the appropriate representation of building `aab:b55112`, and there are different approaches to how the request can be processed. This leads to the various architectural styles discussed in the following sections.

3.2 Validation of representations with respect to a profile

This AS considers that the server can determine whether a representation is conforming to a given profile. Thus, when requested a resource with a profile, it checks which representation validates the provided profile. This can be done in two ways: either by pre-processing or by checking conformance on the fly. We discuss each method separately in the following. Note that these approaches are not mutually exclusive and can therefore be combined.

Pre-processed conformance checking: The server knows a fixed number of profiles that its managed resources may conform to. This way, the server can validate all representations according to each profile in advance, so that it can immediately serve the right representation for a profile without any processing at query time. The pre-process could be done manually when the representations are added to the server, it can be assumed by a guaranteed data generation process, or it can be validated automatically by relying on a schema language and validator.

In our illustrative scenario, a profile could be well known and impose the use of a certain vocabulary (e.g., the BOT ontology) or a profile could consist of a SHACL schema such as the one of Listing 1.1.

Listing 1.1: Example of a profile that consists of a SHACL schema

```
@prefix sh: <http://www.w3.org/ns/shacl#>.
@prefix bot: <https://w3id.org/bot#>.
...
aab:botBuildingShape a sh:NodeShape ;
    sh:targetNode bot:Building ;
    sh:property [ sh:path [ sh:inversePath rdf:type ] ;
                  sh:minCount 1 ; ] .
```

The advantage of this approach is the reduction in response time, but unforeseen constraints are not handled correctly. In this case, the server can either configure a default representation or respond with an error message.

On the fly conformance checking: In this approach, the server is unaware of the concrete constraints that the client is going to send, but assumes that they are structured in a way it can handle. Unlike the previous approach, in order to select the representation to be returned, the server performs a conformance check of the available representations against the client's constraints. This approach is convenient when processing custom constraints, but at the cost of extra time and resource consumption (e.g. computing power). The SHACL shape of Listing 1.1 is an example of what the client may use as a profile, but arbitrary schemas could be submitted that could adhere to Brick or IFC or anything else.

Hybrid conformance checking: To get the best of both worlds, a hybrid approach to compliance checking can be adopted, where the most commonly requested constraints are pre-processed, and in the selection process, it is first checked whether these constraints have already been processed to avoid a new compliance recheck, otherwise it is computed. An additional caching layer could be used to store newly checked constraints to mitigate time and resource losses if the same constraints are requested again.

3.3 Profile compatibility checking

Similarly to media types, profiles may be organised in hierarchies. That is, representations conforming to a profile P_1 may necessarily conform to another profile P_2 because the former is more constraining than the latter. Therefore, when a client requests a profile that does not exactly correspond to a fixed list of profiles supported by the server, it could still be possible to serve the client adequately. In some cases, validating each representation individually against the requested profile is not possible or less advantageous. For example, the requested server may not be the one that hosts the representations and may not have access rights on them. In this case, the server may redirect an authorised client to the right origin server.[10] Also, the validation process may be time consuming for some reason (e.g. large representations and complex constraints to validate).

The alternative approach is a profile compatibility checking, where at the server level, instead of holding the representations, the server has the constraints to which the representation conforms (i.e. a documentation). As such, the selection process shifts from: given a set of representations and a set of constraints, select a representation that maximises a score through a validation checking process (i.e. validate each representation against the set of constraints) to: given a set of documentations (each documentation is a set of constraints which validates a representation) and a set of constraints, select a representation that maximises a measure through a containment checking process (i.e. check the containment of the set of constraints with each of the documentations). In a Semantic Web context, one use could be to check the SHACL shape containment of the requested shapes with the one announced by the server [26]. A more general case is profile containment which takes advantage of profiles linked with the `prof:isProfileOf` property in an RDF graph using the profile vocabulary of the DXWG [5]. Similarly to constraint-to-representation, the checking process could be achieved in a pre-processed, on-the-fly or hybrid manner.

In our scenario UC-A, All-About-Buildings has a large network of collaborators that provide building data to their clients. They restrict access to data to their customers but provide formal specifications of the data profiles. So, the servers of All-About-Buildings can take requests with arbitrary profiles and redirect to the right origin server of a relevant collaborator relying to the profile compatibility check.

Profile compatibility checking can also be preprocessed, on-the-fly, or hybrid.

[10] Origin Server: the server that holds or can generate a resource representation; other types of servers are the so-called intermediaries (e.g. proxy) [17, Section 3].

3.4 Strict (Full) vs Flexible (Partial) conformance

Strict conformance: A profile may define multiple constraints on representations for them to conform. In particular, the more specific features a client demands in its profile preferences, the less a server can satisfy its request. This is because conformance to a profile is typically a binary decision: either a representation *conforms* if it validates all constraints, or *does not conform* if at least one of the constraints is not satisfied. This is what we call *strict* (or *full*) conformance.

To illustrate, a profile defined as a SHACL schema can contain multiple shapes and targets that can be validated individually. In this case, the AS adopts strict conformance when an RDF graph conforms if and only if it validates *all* SHACL constraints.

Flexible conformance: A variation of this style is to rely on a conformance measure to serve partially conformant representations. This measure may equal 1 if all constraints are valid, and 0 if all constraints are invalid.This is what we call flexible (partial) conformance: a server may provide a representation that maximises the satisfaction of the clients' preferences, while not fully conforming to the profile. This is to be compared with the notion of hard vs soft constraints discussed in Section 2. Related, HTTP clients can assign weights to preferences using quality values [17, Section 12.4.2]. Thus, quality values may be considered as a conformance measure by the server.

In our illustrative scenario, a profile may use SHACL with indications of target nodes. A validator can count the number of violations and serve the representation with the least amount of them. Additionally, SHACL can assign a severity to constraints [21, Section 2.1.4], which can be used to modulate conformance measure computation.

The disadvantages of this approach are: that it requires on-the-fly checking, increases the processing requirements, and that the client may not want to get a partially conformant representation. This can be mitigated using additional headers or URI query parameters to let the client specify it requires strict handling of its preferences.

3.5 Unmodified vs Adaptive negotiation

Another way of satisfying a client's request very flexibly is to adapt the response to the client's needs, even when there does not exist a conforming representation on the server's side. In the case when the server does not have any conformant representation, it is sometimes possible to transform a pre-existing one to accommodate the client. In other cases, it may even be possible to completely generate a representation purely based on the request profile. A CN architecture that only provides representations that are hosted on the server is qualified as *unmodified* negotiation, while one that provides a modified or generated representation is *adaptive*.

To illustrate how an adaptive architecture can work, we can assume that the sysadmin of All-About-Buildings inspected the logs of client requests over a period of time, and noticed that a non-negligible number of non-compliant requested profiles use SAREF ontology.[11] Although such profiles are not handled by the company, there is some alignment with available profiles as shown in Listing 1.2.

Listing 1.2: An example of the alignment between BOT and SAREF ontologies

```
@prefix saref4bldg:    <https://saref.etsi.org/saref4bldg/> .
@prefix bot: <https://w3id.org/bot#> .
...
saref4bldg:Building rdfs:subClassOf bot:Building .
saref4bldg:contains rdfs:subPropertyOf bot:containsElement .
```

This procedure has already been used in adaptive CN [30,28,31] for example to adapt content for mobile devices. In a semantic web context this could take the form of RDF graph repairs as addressed in other work [1,2,40]. It could also take other forms (e.g. discarding nodes that violate the constraints, or using alignment techniques to map to a more desired shape). Some additional examples includes: (1) to delete unwanted triples, (2) to use certain units of measurement, (3) to explicit all inferrable triples.

3.6 Local vs third party mediators

CN is not a monolithic process, but rather a layered one with several stages (Discovery, Request Formulation, Selection/Adaptation, Response Indication, Response Interpretation) [45]. This feature allows for two options: either to execute the different stages locally, or to externalise them for execution by third parties. This choice depends among other things on the use case, the characteristic, to gain for example in terms of execution time. In some cases, there can be intermediaries between the server(s) that host the representation(s) and the server that the client interact with. In this case, for example, the discovery process may use a third party component to find other representations of the desired resource. Our scenario UC-B makes use of such architecture and is described further, with implementation details, in Section 5). A third could be: a mediator that stands between the client and the origin server; an auxiliary service that the server relies on to support CN by profile (e.g., a validation service); an auxiliary service that the client relies on (e.g., an adaptation service). With a third party, it is possible to emulate CN by profile even if the origin server only implements regular CN.

In our scenario UC-A, we assumed initially that the company hosts different representations in RDF that use different ontologies. As the ontologies for buildings and related data diversify, the company may realise that it could keep a single representation and externalise adaptation to a service that is dedicated to keeping alignments between vocabularies up to date, with associated transformation functions. Obviously, the drawback of this approach is the lack of control over the external component, as well as potential latency issues.

[11] SAREF Extension for Building: https://saref.etsi.org/saref4bldg/

3.7 Discrete vs Non-discrete representations

A website that only provides representations for a fixed set of URIs can have the complete state of its resources transferred as files organised in folders, even though these may be generated from a monolithic database. This is what we call discrete representations. In other cases, the server can handle an open set of URIs that are composed by the client using query parameters. In this case, one cannot get the complete state of the website as files organised in folders following the path structure of the URIs, because the number of URIs handled by the server is arbitrarily large.[12] This is what we call non-discrete representations. Note that this can be viewed as a CN AS because it allows one to use URI-based CN (as seen in Section 2) for fine-grained CN by profile. A server having on non-discrete representations normally relies on a database management system that group all data together, and should not host the representations per se, except perhaps for caching purposes.

In our running scenario, if we assume the company opted for an external adaptation service, it no longer needs to keep even a single canonical representation for each resource. Instead, it can extract relevant pieces from the database and post them to the adaptation service that will compose the desired representation. Note that this would only work on restricted cases that the adaptation service is able to handle, with well specified inputs and outputs.

The advantage of discrete representation is that it is easy to set up and easy to retrieve. The complete website can be cached if there is not fast-changing data. However, it is not practical in certain use cases, especially when there is a high level of customisability by the user (i.e. client). In a Semantic Web context, when negotiating in the profile dimension, we can consider both approaches. Some well-identified resources may be served from separate files (that may be "virtual", resulting from a generation script) but an arbitrary set of other representations may be the result of request-specific construction. If the origin server provides a SPARQL endpoint but is not able to do CN by profile, a tool that translates SHACL to SPARQL [15] could be used to achieve CN in this context.

3.8 Centralised vs Distributed representations

In traditional CN, different URIs that dereference to different representations can be assumed to denote different resources. If we take this assumption (which is the *unique name assumption* or UNA), it is never necessary to request two URIs to obtain the variants of a single resource. However, there is no formal obligation that UNA holds on the Web, and Semantic Web languages allow several URIs to denote the same resource. For example, to identify the white spruce (Picea glauca) species, different servers use different URIs: U_1: `https://www.uniprot.org/taxonomy/3330`, U_2: `http://dbpedia.org/resource/Picea_glauca` or even U_3: `http://www.wikidata.org/entity/Q128116`, to name a few. Each of these servers has a subset of the available representations of the resource, meaning

[12] Though never infinite, given the restrictions on URI length.

that when negotiating with individual servers we are considering only a subset of the potentially valid representations.

Taking into account the potential distribution of representations for a resource leads to architectures for CN that must deal with multiple origin servers in one request. The first step is to axiomatise that, e.g. U_1, U_2 and U_3 actually identify the same resource. This can be done using Semantic Web technologies, such as equivalence links (e.g. `owl:sameAs` or alternatively (1) `skos:exactMatch` ands `skos:closeMatch` from the ontology [32] (2) `wdt:P2888` by Wikidata[13] (3) `umbel:isLike` in *Upper Mapping and Binding Exchange Layer ontology*[14] (4) `schema:sameAs` in http://schema.org/. (5) and finally the predicates introduced in the *similarity ontology* [19]. The sameAs portal[15] provides a service to find equivalence links after extracting them from different sources. The second step is to build a portal that is aware of this distribution and is able to negotiate with the potential servers and aggregate the results. While the distributed architecture allows the discovery of additional potential representations, it comes at the cost of negotiating with multiple servers and is consequently time consuming.

4 Synthesis of architectural style survey

In this section, we provide a summary of the different AS discussed in Section 3 with their advantages and disadvantages, all reported in Table 1. We additionally provide, for each AS, references to publications or implementations that make use of said style (see the *Reference* column). Such information provides context and exemplifies the different AS.

Further, we here consider the implementations [13,12,14,4] collected in the W3C content negotiation by profile implementation report [43]. They all use the *Unmodified*, *Local* and *Strict* AS. *Media Types Register* [13] uses a *Non-Discrete* AS (all media types are stored in a `mediatypes.ttl` file and SPARQL queries are used for the extraction) while the others use a *Discrete* AS. *CKAN DCAT Plugin* [14] proposes harvester plugins to allow automatic import of datasets from remote sources, thus we judge that it uses a *Decentralised* AS while the others use a *Centralised* AS. As for the last two AS, all implementations *Pre-process* the validation. However, [12,13] are both *Profile-compatibility-checking* AS: first they validate the requested profile against the available ones, then they generate the representations with template rendering. Conversely, [4] uses the ProfileWiz tool [6] to generate profiled representations that are used in the CN by profile, making it *Representation-validation* AS.

[13] In Wikidata the property P2888 is “Exact match”: https://www.wikidata.org/wiki/Property:P2888

[14] Umbel: https://lov.linkeddata.es/dataset/lov/vocabs/umbel

[15] SameAs portal: http://sameas.org/

AS	AS Specific	Advantage	Disadvantage	References
Representation validation	Pre-processed validation	• Low response time • Cacheable response	• Unable to validate unforeseen constraints	[4,48,14]
	On the fly validation	• Handle unforeseen constraints	• Additional conformance check time • Additional resource consumption (e.g. computing power) • Caching response is not pertinent	[45]
	Hybrid approach	• Low response time • Partial cacheable response • Handle unforeseen constraints	• Additional configuration required to accept hybrid approach • Additional resource consumption (e.g. computing power)	[30,31]
Profile compatibility checking	Pre-processed	• Low response time • Cacheable response	• Unable to validate unforeseen constraints	[12,13]
	On the fly	• Handle unforeseen constraints	• Additional conformance check time • Additional resource consumption (e.g. computing power) • Caching response is not pertinent	[11]
	Hybrid approach	• Low response time • Partial cacheable response • Handle unforeseen constraints	• Additional configuration required to accept hybrid approach • Additional resource consumption (e.g. computing power)	
Strict (Full) vs Flexible (Partial) conformance	Strict (Full)	• Low response time • Cacheable response • Compatible with Pre-processed validation, on the fly validation and their hybrid AS	• Only preconfigured default representation is possible • Partial and total conformance of constraints is treated equally • Constraints precedence is not considered	[13,12,4,48,27,31,14]
	Flexible (Partial)	• A representation is served even if it partially conforms to the constraints • Constraints precedence is considered	• Only compatible with on-the-fly AS • Additional conformance check time • Additional resource consumption (e.g. computing power) • Caching response is not pertinent	[11,45]
Unmodified vs Adaptive negotiation	Unmodified	• Low response time • Cacheable • response	• with unforeseen constraints only error or default response is sent	[11,13,12,4,45,48,14]
	Adaptive	• Adapt available representation to unforeseen constraints	• Additional adaptation process time • Additional resource consumption (e.g. computing power) • Caching response is not pertinent	[30,29,3,27,31]
Local vs 3rd party mediators	Local	• Low response time • Full control on the executed process	• Must implement and manage the process locally • More resource consumption (e.g. computing power)	[11,13,12,4,45,48,3,27,14]
	3rd party mediators	• The implementation and management of the process is delegated • Less resource consumption (e.g. computing power	• Less control and extends all the risks of the 3rd party • Additional time (e.g. transfer and authentication) • Need to Coordinate 3rd parties	[31]
	Hybrid approach	• The implementation and management of critical processes is done locally • Extends the advantages of (local/3rd party) when used	• Extends the disadvantages of (local/3rd party) when used	[30,27]
Discrete vs Non-discrete Representations	Discrete	• Easy to set up, the default AS for some servers • Easy to retrieve the representation	• Must manage the duplicate representation changes in complex representations	[11,12,4,45,48,30,3,27,31,14]
	Non-discrete	• One source of truth for complex representations	• Must be configured and manage the retrieval of representations	[13]
Centralised vs Distributed CN	Centralised	• All representations are available in one place	• Not practical for the Web scale	[11,13,12,4,45,48,3,27,31]
	Distributed	• Practical for the Web scale • More representations are available	• Less control since the CN is externalised to other servers • Additional time of the distribution request and restitution of response	[14]

Table 1: Content Negotiation Architectural Style Review

5 Implementation of UC-B

In this section we focus on UC-B for which we provide an implementation to address its requirements.[16] The use case is highlighting a combination of AS that distinguishes itself from the existing approaches already available, as can be seen in Table 1.

In more details, the use case is the following: a server provides RDF representations of a large portion of the Linked Open Data cloud by way of a URI scheme that can identify any resource on the Web. The server does not host any data itself but fetch them from the Web dynamically at request time. In order to identify which origin server it must contact, it relies on a third party service called *sameAs*[17]. This service has an API that takes a URI as input and outputs a list of URIs that are assumed to denote the same resource according to the service. The URIs in the list are then sequentially requested for validation against a SHACL schema. The general architecture is shown in Figure 2.

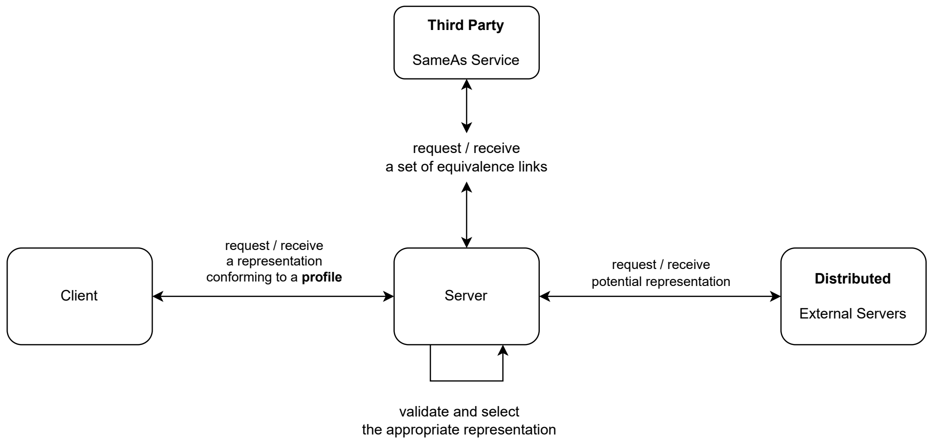

Fig. 2: Overview of the architecture for UC-B

The implementation is developed in Java and is openly available. It is built using the *Spring framework* (version 2.7.5). To test it, we provide an endpoint at *</dcn/api/profile>*. The client can request any URI of the Web by passing it in the `iri` query parameter. The expected profile is conveyed in the header `Accept-profile` in the form of a URI that must dereference to a SHACL schema. The SHACL file may be located in a remote web server. *Swagger* (version 3.0.0) is used to provide a friendly API documentation.

The negotiation starts with our server trying to check if there is an appropriate representation directly from the location of the URI given in parameter.

[16] Anonymised repository: https://anonymous.4open.science/r/anonym-1C65
[17] SameAs service home page: http://sameas.org/

A validation occurs on RDF data. If it validates, the representation is returned and nothing else happens. If not, our server then looks up a set of sameAs URIs. Then it goes one step further to check if any of the discovered alternatives are valid. If no representation could be found, only then will the server respond with an error *406 (NOT ACCEPTABLE)* [17, Section 15.5.7]. The SHACL validation occurs locally on our server but we must first fetch the SHACL shape from the URI given in `Accept-profile`. *Apache Jena* (version 4.4.0) is used to manipulate the RDF graphs and perform the validation. The server keeps track of the best graph and uses the *isABetterRepresentationThan* function to test whether an alternative valid graph is better than the current best graph.

If we analyse our implementation to extract the used AS, we can observe that we fall into *Representation-validation* AS with *On-the-Fly* validation. Note that there is no caching enabled in the current version, if such a caching mechanism existed we would fall into the *hybrid validation* AS. We do not adapt the RDF sources if they do not conform to the requested profile, which makes it fall into the *Unmodified* AS. Since the alternative representations are found through the third party sameAs portal and the validation is done locally, this makes it *Hybrid* in this AS axis. At the same time, since the representations are not on a central server, this makes the implementation *Distributed*. In such a case, no prior assumption could be made about the way the representations are stored (i.e. either *Discrete* or *Non-Discrete* AS could be used). Finally, the implementation takes into account the severity of the constraints, either in the validation or in *isBetterRepresentationThan*, to select the best representation to serve, thus making it conform to the *Flexible Conformance* AS.

6 Conclusion

In this paper, we give an overview of the possible AS in CN and their application in a Semantic Web context using semantic validation, taking RDF graphs as representations and SHACL as validation language. We illustrate some of these AS through a use case and an implementation to show their practicality. We found that a large number of architectural options are available for CN by profile, and only a handful of the possible combinations have been implemented and tested. A real-life experimental comparison of diverse AS remains to be conducted in terms of performance, complexity, and availability of representations. This work also opens grounds for more investigation on advanced content negotiation, a fundamental aspect of the Web that has been somewhat neglected in research.

Supplemental Material Statement: Source code is available from *anonymous 4open science.*[18].

References

1. Ahmetaj, S., David, R., Ortiz, M., Polleres, A., Shehu, B., Šimkus, M.: Reasoning about Explanations for Non-validation in SHACL. In: Proceedings of the 18th In-

[18] https://anonymous.4open.science/r/anonym-1C65

ternational Conference on Principles of Knowledge Representation and Reasoning. pp. 12–21 (Nov 2021), https://doi.org/10.24963/kr.2021/2

2. Ahmetaj, S., David, R., Polleres, A., Simkus, M.: Repairing SHACL Constraint Violations Using Answer Set Programming. In: The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13489, pp. 375–391. Springer (Oct 2022), https://doi.org/10.1007/978-3-031-19433-7_22
3. Ankolekar, A., Vrandecic, D.: Personalizing web surfing with semantically enriched peronal profiles. In: Proceedings of the Semantic Web Personalization Workshop (Jun 2006), https://www.aifb.kit.edu/images/2/26/2006_1201_Ankolekar_Personalizing_w_1.pdf
4. Atkinson, R.: OGC Definitions Server (2020), https://www.ogc.org/resources/def-server/
5. Atkinson, R., Car, N.J.: The Profiles Vocabulary. W3C Working Group Note, W3C (Dec 18 2019), https://www.w3.org/TR/2019/NOTE-dx-prof-20191218/
6. Atkinson, R., Car, N.J.: The Profile Wiz Tool (2021), https://github.com/RDFLib/profilewiz
7. Berners-Lee, T., Cailliau, R., Groff, J.F., Pollermann, B.: World-Wide Web: The Information Universe. Electronic Networking: Research, Applications and Policy **2**(1), 74–82 (Jan 1 1992), https://doi.org/10.1108/eb047254
8. Berners-Lee, T., Fielding, R.T., Masinter, L.: Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, Internet Engineering Task Force (Jan 2005), http://tools.ietf.org/html/rfc3986
9. Berrueta, D., Phipps, J., Miles, A., Baker, T., Swick, R.: Best Practice Recipes for Publishing RDF Vocabularies. W3C Working Group Note, W3C (Aug 28 2008), https://www.w3.org/TR/swbp-vocab-pub/
10. van den Brink, L., Barnaghi, P.M., Tandy, J., Atemezing, G., Atkinson, R., Cochrane, B., Fathy, Y., García-Castro, R., Haller, A., Harth, A., Janowicz, K., Kolozali, S., van Leeuwen, B., Lefrançois, M., Lieberman, J., Perego, A., Le Phuoc, D., Roberts, B., Taylor, K., Troncy, R.: Best practices for publishing, retrieving, and using spatial data on the web. Semantic Web Journal **10**(1), 95–114 (2019), https://doi.org/10.3233/SW-180305
11. Butler, M.H.: Implementing Content Negotiation using CC/PP and WAP UAProf. Tech. Rep. HPL-2001-190, HP Lab (2001), https://www.hpl.hp.com/techreports/2001/HPL-2001-190.html
12. Car, N.J.: PHP ConnegP (2019), https://github.com/nicholascar/php-connegp/, (Version 0.7)
13. Car, N.J.: Media Types Register (2021), https://github.com/nicholascar/mediatypes-service
14. Car, N.J.: CKAN DCAT Plugin (2022), https://github.com/ckan/ckanext-dcat, (Version 1.4.0)
15. Corman, J., Florenzano, F., Reutter, J.L., Savkovic, O.: SHACL2SPARQL: Validating a SPARQL Endpoint against Recursive SHACL Constraints. In: Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas), Auckland, New Zealand, October 26-30, 2019. CEUR Workshop Proceedings, vol. 2456, pp. 165–168. CEUR-WS.org (Oct 2019), https://ceur-ws.org/Vol-2456/paper43.pdf
16. Farias Lóscio, B., Burle, C., Calegari, N.: Data on the Web Best Practices. W3C Recommendation, W3C (Jan 31 2017), https://www.w3.org/TR/2017/REC-dwbp-20170131/

17. Fielding, R.T., Nottingham, M., Reschke, J.F.: HTTP Semantics. RFC 9110, IETF (Jun 2022), http://www.ietf.org/rfc/rfc9110.txt
18. Freed, N., Klensin, J.C., Hansen, T.: Media Type Specifications and Registration Procedures. RFC 6838, IETF (Jan 2013), https://doi.org/10.17487/RFC6838
19. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data. In: The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I. pp. 305–320 (Nov 2010), https://doi.org/10.1007/978-3-642-17746-0_20
20. Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M.H., Tran, L.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation, W3C (Jan 15 2004), https://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/
21. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL). W3C Recommendation, W3C (Jul 20 2017), https://www.w3.org/TR/2017/REC-shacl-20170720/
22. Kunze, J.A., Bermès, E.: The ARK Identifier Scheme. Internet-Draft draft-kunze-ark-36, Internet Engineering Task Force (Oct 17 2022), https://datatracker.ietf.org/doc/draft-kunze-ark/36/, work in Progress
23. Lagoze, C., Van de Sompel, H., Nelson, M., Warner, S.: The Open Archives Initiative Protocol for Metadata Harvesting. Tech. rep., Open Archives Initiative (Jun 14 2002), http://www.openarchives.org/OAI/openarchivesprotocol.html, (Version 2.0)
24. Lanthaler, M.: The Profile URI Registry. RFC 7284, IETF (2014)
25. Lefrançois, M.: RDF presentation and correct content conveyance for legacy services and the web of things. In: Janowicz, K., Kuhn, W., Cena, F., Haller, A., Vamvoudakis, K.G. (eds.) Proceedings of the 8th International Conference on the Internet of Things, IOT 2018, Santa Barbara, CA, USA, October 15-18, 2018. pp. 43:1–43:8. ACM Press (Oct 2018), https://doi.org/10.1145/3277593.3277618
26. Leinberger, M., Seifer, P., Rienstra, T., Lämmel, R., Staab, S.: Deciding SHACL Shape Containment Through Description Logics Reasoning. In: The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12506, pp. 366–383. Springer (Nov 2020)
27. Lemlouma, T., Layaïda, N.: NAC: A Basic Core for the Adaptation and Negotiation of Multimedia Services. Opera Project, INRIA (2001), https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3a48e0bd1d62b59f53231090d7808494a421c572
28. Lemlouma, T., Layaïda, N.: Universal Profiling for Content Negotiation and Adaptation in Heterogeneous Environments. In: W3C Workshop on Delivery Context. pp. 4–5 (2002), https://www.w3.org/2002/02/DIWS/submission/tlemlouma-W3CPositionPaper03-2002.htm
29. Lerouge, S.: Personalizing quality aspects for video communication in constrained heterogeneous environments. Ph.D. thesis, Ghent University (Nov 24 2006), https://backoffice.biblio.ugent.be/download/467760/1878996
30. Lum, W.Y., Lau, F.C.M.: On Balancing between Transcoding Overhead and Spatial Consumption in Content Adaptation. In: Akyildiz, I.F., Lin, J.Y.B., Jain, R., Bharghavan, V., Campbell, A.T. (eds.) Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking, MOBICOM 2002, Atlanta, Georgia, USA, September 23-28, 2002. pp. 239–250. ACM (Sep 2002), https://doi.org/10.1145/570645.570675

31. Lum, W.Y., Lau, F.C.M.: User-Centric Content Negotiation for Effective Adaptation Service in Mobile Computing. IEEE Transaction on Software Engineering **29**(12), 1100–1111 (Dec 2003), https://doi.org/10.1109/TSE.2003.1265524
32. Miles, A., Bechhofer, S.: SKOS Simple Knowledge Organization System, Reference, W3C Recommendation 18 August 2009. W3C Recommendation, W3C (Aug 18 2009), http://www.w3.org/TR/2009/REC-skos-reference-20090818/
33. Murata, M., Laurent, S.S., Kohn, D.: XML Media Types. RFC 3023, Network Working Group (Jan 2001), https://doi.org/10.17487/RFC3023
34. Nebert, D., Voges, U., Vretanos, P., Bigagli, L., Westcott, B.: OGC® Catalogue Services 3.0 Specification - HTTP Protocol Binding. Tech. rep., Open Geospatial Consortium (Jun 10 2016), https://docs.ogc.org/is/12-176r7/12-176r7.html
35. Nottingham, M.: Web Linking. RFC 8288, IETF (Oct 2017), https://doi.org/10.17487/RFC8288
36. Ohto, H., Hjelm, J.: CC/PP exchange protocol based on HTTP Extension Framework. W3c note, W3C (Jun 24 1999), https://www.w3.org/TR/NOTE-CCPPexchange
37. Pullmann, J., Atkinson, R., Isaac, A., Faniel, I.: Dataset Exchange Use Cases and Requirements. W3C Working Group Note, W3C (Jan 17 2019), https://www.w3.org/TR/2019/NOTE-dcat-ucr-20190117
38. Romary, L., Lundberg, S.: The 'application/tei+xml' Media Type. RFC 6129, IETF (Feb 2011), https://doi.org/10.17487/RFC6129
39. Snell, J.: Prefer Header for HTTP. RFC 7240, IETF (Jun 2014), https://www.rfc-editor.org/rfc/rfc7240
40. Solanki, M., Mader, C.: Towards Maintainable Constraint Validation and Repair for Taxonomies: The PoolParty Approach. In: Proceedings of the 7th International Workshop on Consuming Linked Data co-located with 15th International Semantic Web Conference, COLD@ISWC 2015, Kobe, Japan, October 18, 2016. CEUR Workshop Proceedings, vol. 1666. CEUR-WS.org (Oct 2016), http://ceur-ws.org/Vol-1666/paper-06.pdf
41. Svensson, L.: An http Header for Metadata Schema Negotiation. In: W3C Workshop on Smart Descriptions & Smarter Vocabularies (SDSVoc). W3C (Nov 2016)
42. Svensson, L.G., Atkinson, R., Car, N.J.: Content Negotiation by Profile. W3C Working Draft, W3C (Nov 26 2019), https://www.w3.org/TR/2019/WD-dx-prof-conneg-20191126/
43. Svensson, L.G., Atkinson, R., Car, N.J.: Content Negotiation by Profile Implementation Report. W3C-internal document, W3C (Mar 21 2022), https://w3c.github.io/dx-connegp/connegp-implementation-report/
44. Svensson, L.G., Verborgh, R., Van de Sompel, H.: Indicating, Discovering, Negotiating, and Writing Profiled Representations. Internet-draft, Internet Engineering Task Force (Mar 9 2021), https://www.ietf.org/archive/id/draft-svensson-profiled-representations-01.txt
45. Taghzouti, Y., Vachtsevanou, D., Mayer, S., Ciortea, A.: A Step Toward Semantic Content Negotiation. In: Companion Proceedings of the 23rd International Conference on Knowledge Engineering and Knowledge Management, Bozen-Bolzano, Italy, September 26-29, 2022. CEUR Workshop Proceedings, vol. 3256. CEUR-WS.org (Sep 2022), https://ceur-ws.org/Vol-3256/paper5.pdf
46. Taghzouti, Y., Zimmermann, A., Lefrançois, M.: Content Negotiation on the Web: State of the Art. Tech. Rep. abs/2204.10097, CoRR (Apr 21 2022), https://arxiv.org/abs/2204.10097

47. Verborgh, R.: Your JSON Is Not My JSON – A Case for More Fine-Grained Content Negotiation. In: Proceedings of the Workshop on Smart Descriptions & Smarter Vocabularies (Nov 2016), https://ruben.verborgh.org/articles/fine-grained-content-negotiation/
48. Viot, P., Thouvenin, N.: ISTEX: Une Nouvelle Corde à Son ARK. Arabesques **88**, 18–19 (2018), https://publications-prairial.fr/arabesques/index.php?id=1222
49. Wilde, E.: The 'profile' Link Relation Type. RFC 6906, Independent Submission (Mar 2013), https://doi.org/10.17487/RFC6906