Protocols and architecture

SEPA training

24th Conference of the Open Innovations Association FRUCT

Moscow, Russia

April 9th-10th, 2019

Luca Roffia

Research follow, Adjunct Professor

Department of Computer Science and Engineering, University of Bologna

luca.roffia@unibo.it

https://site.unibo.it/wot/en

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
ARCES

IEEE

IEEE ComSoc
IEEE Communications Society

IEEE
Italy Section

Standards: protocols and formats

SPARQL 1.1 **Query** Language
https://www.w3.org/TR/sparql11-query/

SPARQL 1.1 **Federated** Query
https://www.w3.org/TR/sparql11-federated-query/

W3C

(**2013**)

SPARQL 1.1 **Update** Language
https://www.w3.org/TR/sparql11-update/

SPARQL 1.1 **Graph Store** HTTP Protocol
https://www.w3.org/TR/sparql11-http-rdf-update/

SPARQL 1.1 Protocol
https://www.w3.org/TR/sparql11-protocol/

SPARQL 1.1 endpoints

amazon

blazegraph

OPENLINK SOFTWARE
Making Technology Work For You®

THE APACHE SOFTWARE FOUNDATION

Linked Data Platform 1.0 (2015)
https://www.w3.org/TR/ldp/

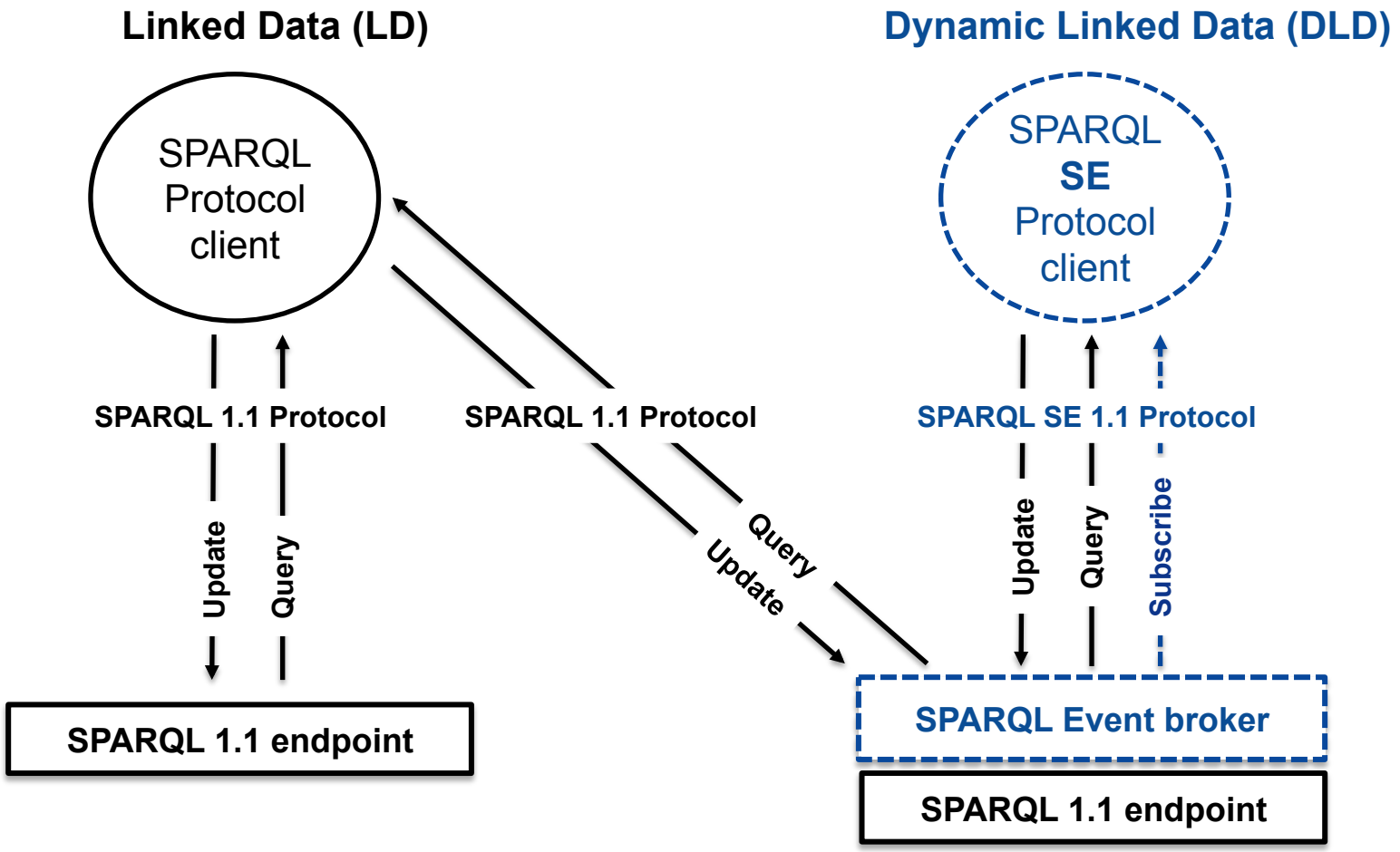
Linked Data Notifications (2017)
https://www.w3.org/TR/ldn/

JSON-LD 1.0 (2014)
https://www.w3.org/TR/json-ld/

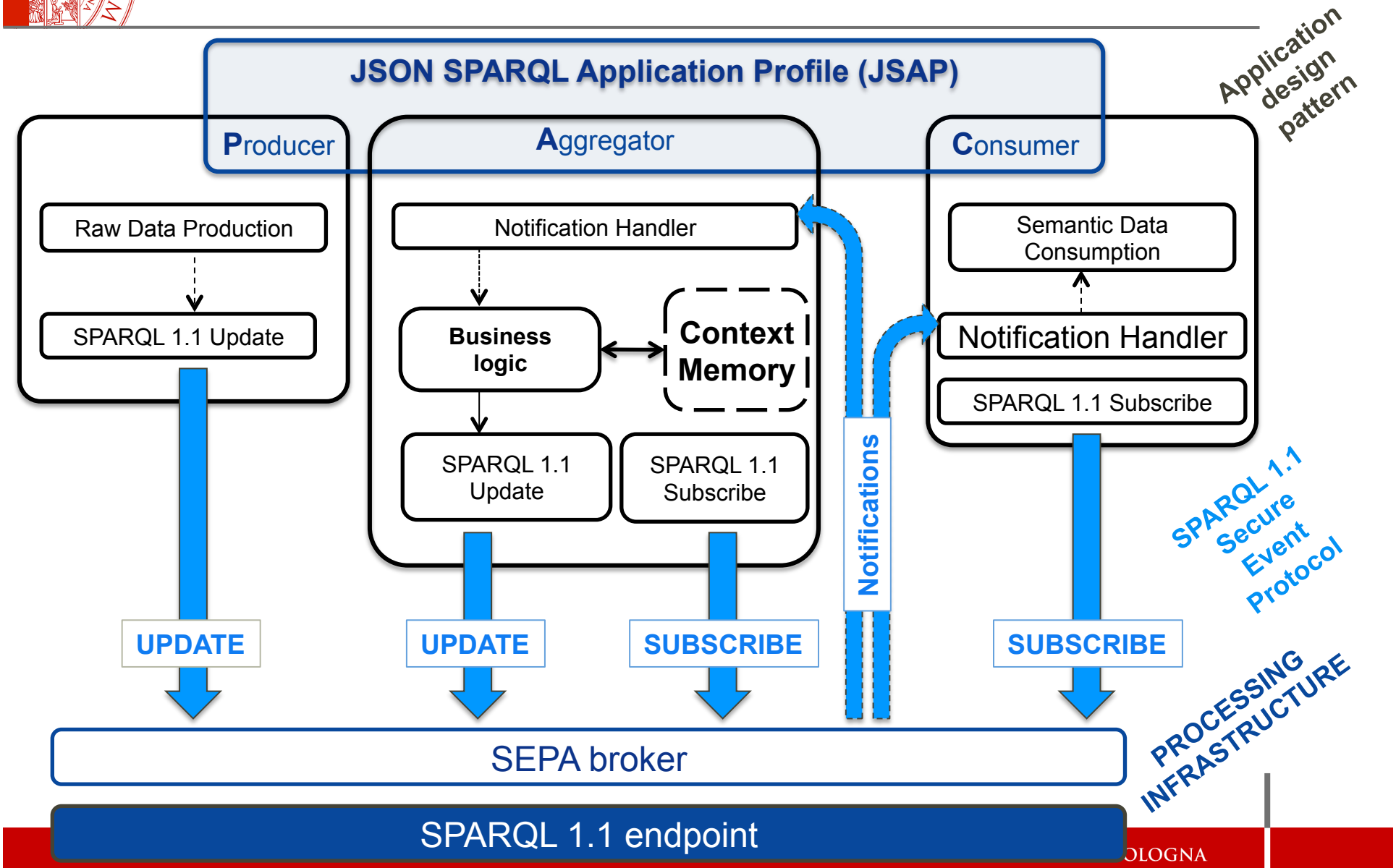
RDF 1.1 (2014)
https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

RDFS 1.1 (2014)
https://www.w3.org/TR/2014/REC-rdf-schema-20140225/

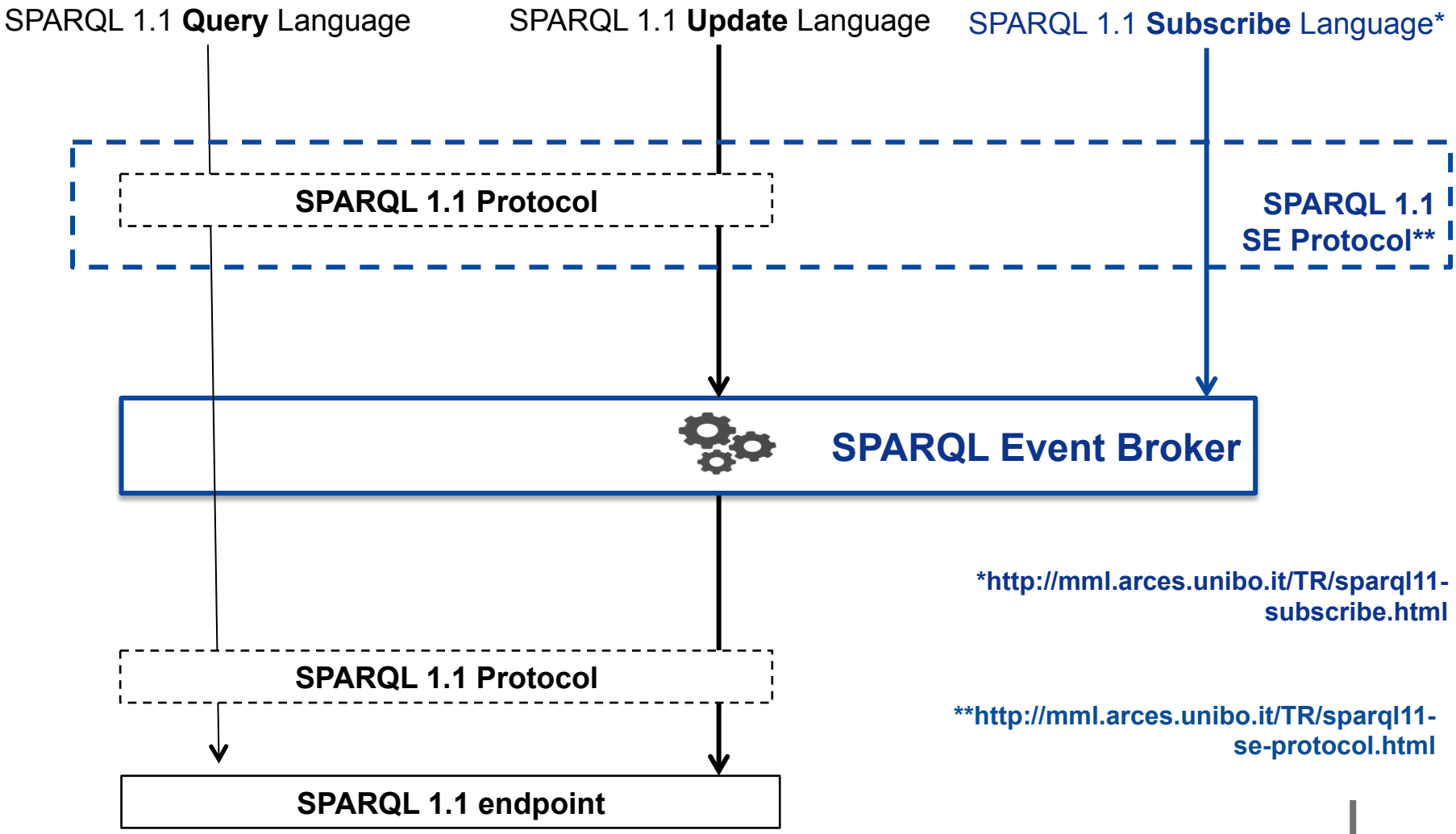
From LD to DLD



SPARQL Event Processing Architecture



SPARQL Event Broker

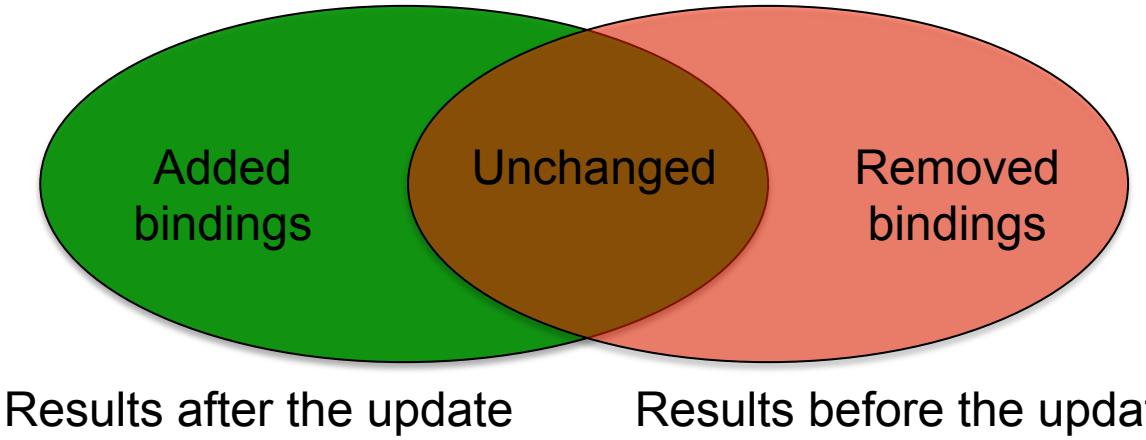


*http://mml.arces.unibo.it/TR/sparql11-subscribe.html

**http://mml.arces.unibo.it/TR/sparql11-se-protocol.html

Publish-subscribe mechanism

- content based (e.g., MQTT is topic-based)
- events are generated by means of SPARQL 1.1 Update language
- subscription to events is expressed using the SPARQL 1.1 Query language
- at subscription time the query results are returned
- a notification includes the results that have been changed (i.e., added and removed bindings)



SPARQL 1.1 Secure Event Protocol

- A SEPA implementation is RECOMMENDED to :
  - use **WebSocket protocol** [RFC6455] for the subscription mechanism
  - express subscribe/unsubscribe requests and notification according to the **SPARQL 1.1 Subscribe language**
  - provide **data encryption**, **server authentication** and **message integrity** according to **TLS** [RFC5246]
  - support **HTTPS** [RFC2818] and **WSS** [RFC6455] protocols
  - be compliant with the security mechanism RECOMMENDED by the **Web of Things** interest group.

SPARQL 1.1 Subscribe: request

```
{"subscribe" : {
  "sparql" : "select * where {?vaimee ?deda ?didi}",
  "authorization" : "Bearer xabtQWoH8RJJk1FyK2iEs9asA=",
  "alias" : "All",
  "default-graph-uri" : "http://example/uri_of_the_default_graph",
  "named-graph-uri" : "http://example/uri_of_a_named_graph"
 }}
```

sparql (**required**) :SPARQL 1.1 query

authorization (optional): JSON Web Token [RFC7519]

alias (optional): if specified, it would be included in every corresponding notifications. The reason behind the alias is that if multiple subscribe requests are sent in parallel, the order of the first notifications is not guarantee.

default-graph-uri/named-graph-uri (optional): allow specifying an RDF Dataset as state in Section 2.1.4 of the SPARQL 1.1 protocol specification. “A SPARQL query is executed against an RDF Dataset. The RDF Dataset for a query may be specified either via the default-graph-uri and named-graph-uri parameters in the SPARQL Protocol or in the SPARQL query string using the FROM and FROM NAMED keywords.”

SPARQL 1.1 Subscribe: notification


```
{"notification":{
  "spuid" : "sepa://subscription/0d057ca5-cc10-4e8a-a5d9-59d7b36f71d6",
  "sequence" : 0,
  "alias" : "All",
  "addedResults" :{
   "head": { "vars": ["vaimee","deda
   "results": {
    "bindings": [{
     "vaimee": {"type": "uri", "value"
     "deda": {"type": "uri", "value": "
     "didi": {"type": "literal","value": '
  "removedResults":{}}}
```



```
{"notification":{
  "spuid" : "sepa://subscription/0d057ca5-cc10-4e8a-
a5d9-59d7b36f71d6",
  "sequence" : 1,
  "alias" : "All",
  "addedResults" :{
   "head": { "vars": ["vaimee","deda","didi"] },
   "results": {
    "bindings": [{
     "didi": {"type": "literal","value": "ზპკლ33"} }]}},
  "removedResults":{
   "head": { "vars": ["vaimee","deda","didi"] },
   "results": {
    "bindings": [{
     "didi": {"type": "literal","value": "ვაიმეე"} }]}}}
```


spuid (**required**): a unique URI

sequence (**required**): always 0 on the first notification, incremented by one on each notification.

addedResults/**removedResults** **(required)**: are expressed according to SPARQL 1.1 Query Results JSON Format https://www.w3.org/TR/sparql11-results-json/

SPARQL 1.1 Subscribe: unsubscribe and error

Request

```
{"unsubscribe":{
   "spuid":"sepa://subscription/0d057ca5-cc10-4e8a-a5d9-59d7b36f71d6",
   "authorization" :"Bearer eyJhbGciOiJIUzI....ONFh7HgQ"}}
```

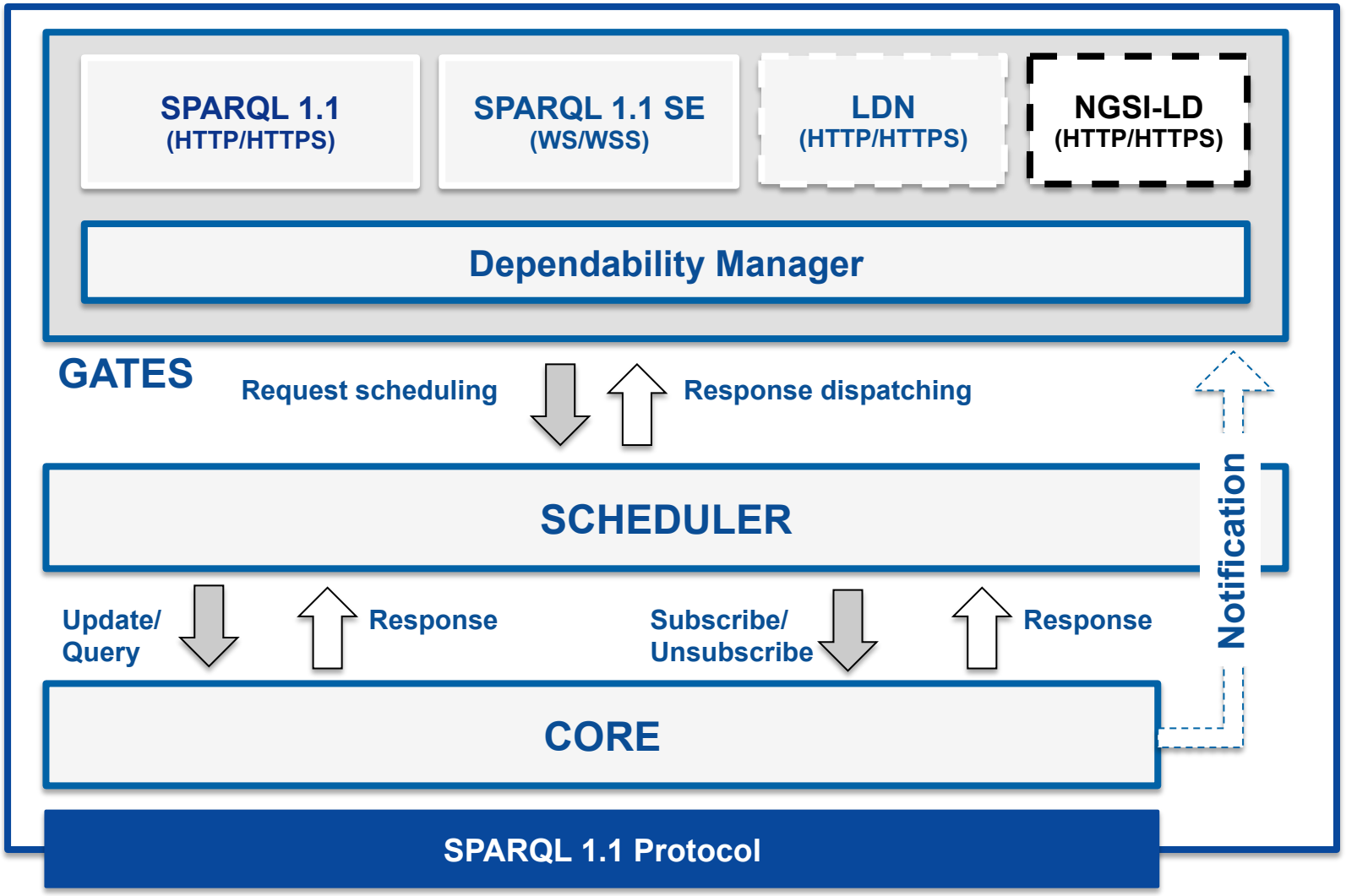
Response

```
{"unsubscribed":{
   "spuid":"sepa://subscription/0d057ca5-cc10-4e8a-a5d9-59d7b36f71d6"}}
```

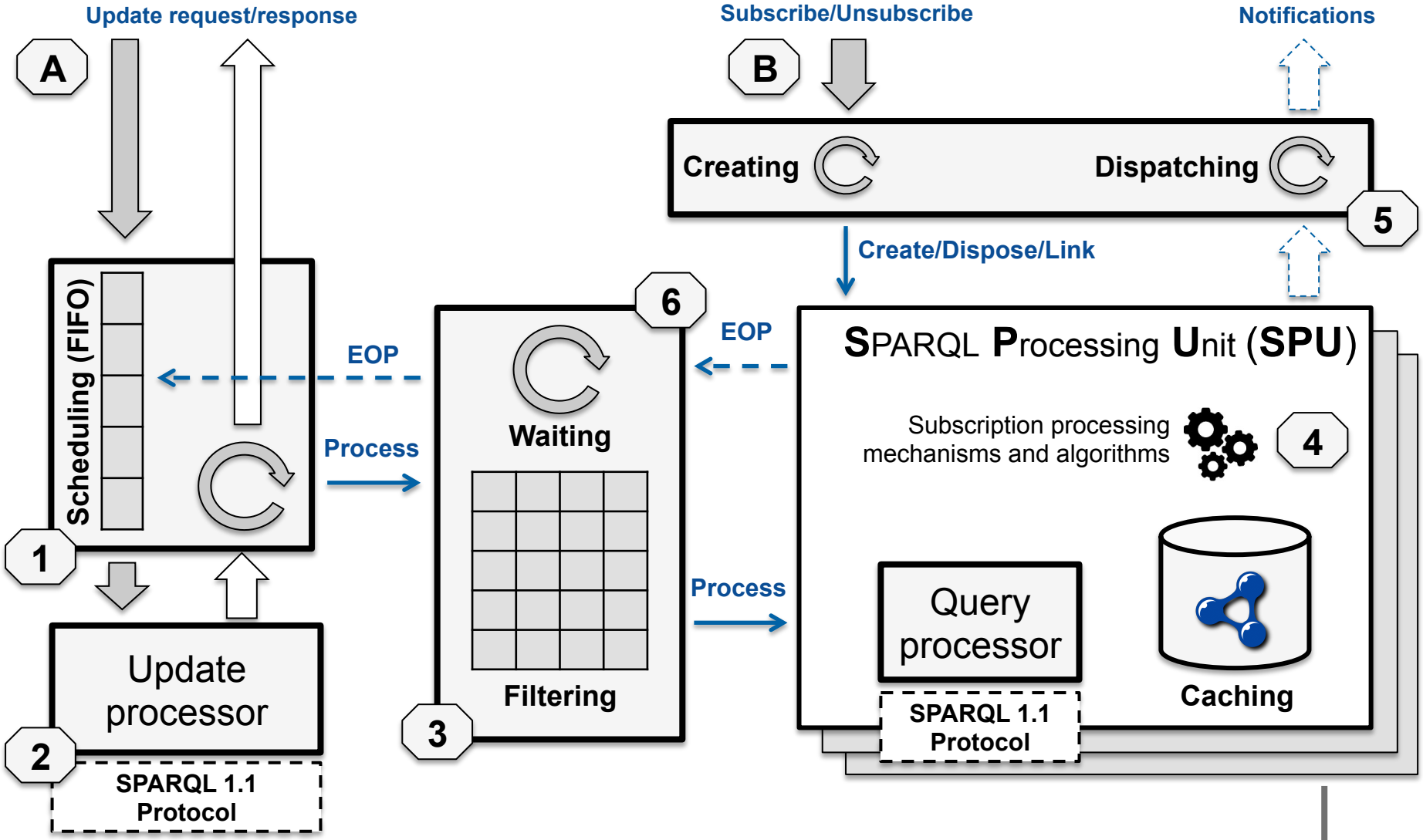
Error response

```
{
  "error":"unauthorized_client",
  "error_description" : "Client is not authorized",
  "status_code" : 401
}
```

SEPA broker architecture



Broker CORE architecture



Development directions

- NGSI-LD Context Management APIs (ETSI standard)
- Subscription algorithms
  - Update processing (FRUCT 23 poster)
  - Filtering (LUTT)
  - Caching (how to build the a local context)
  - Rete
- Distributed architecture
- Web of Things discovery and interaction framework (Cocktail)