



WS/HTTP-DDS

Accessing Real-Time DDS Data From
Web-Based Clients

Andrea Iannitti

Fabrizio Bertocci

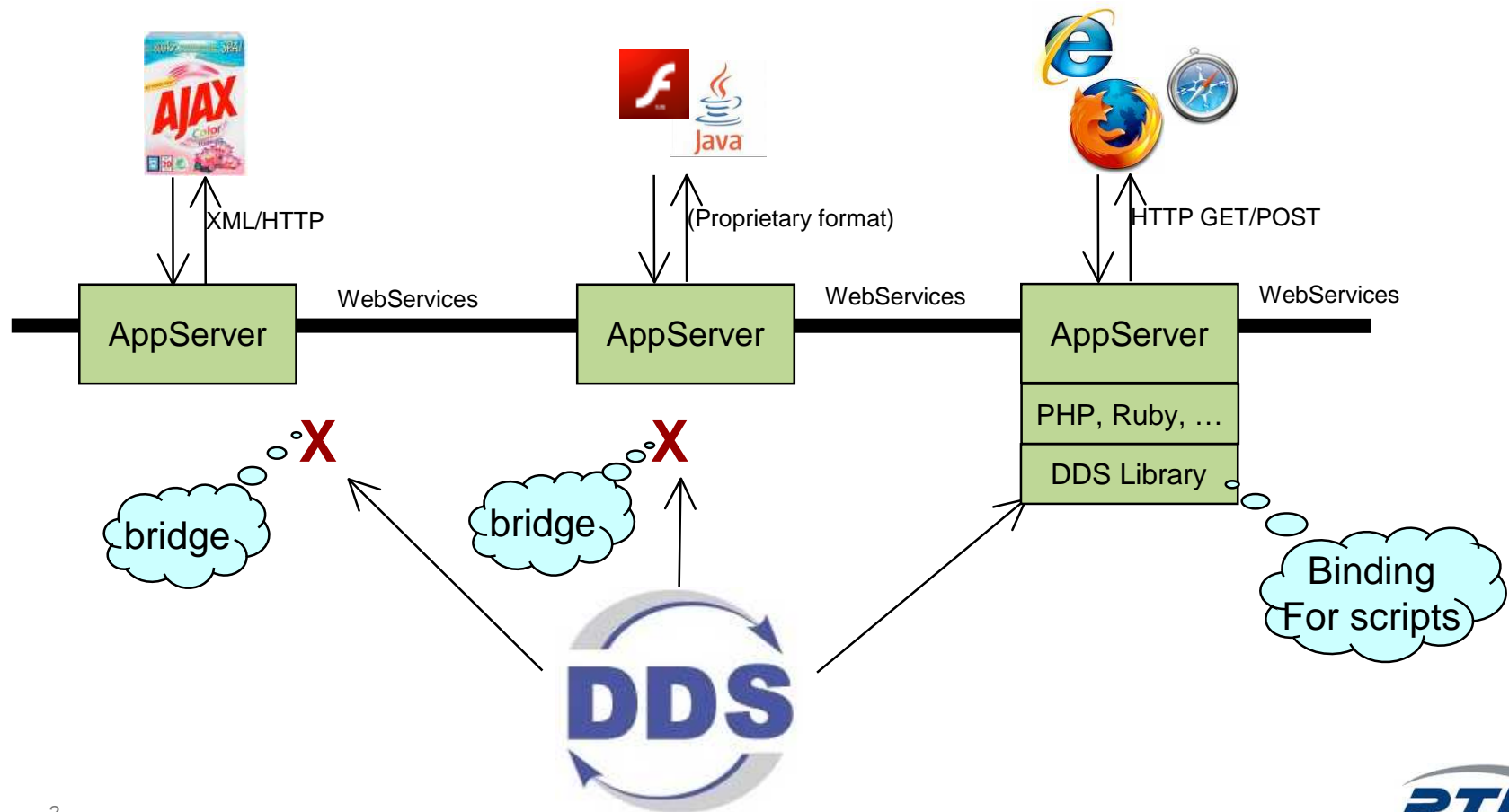
Gerardo Pardo, Ph.D.

Nick Stavros, Ph.D.

July 14, 2008

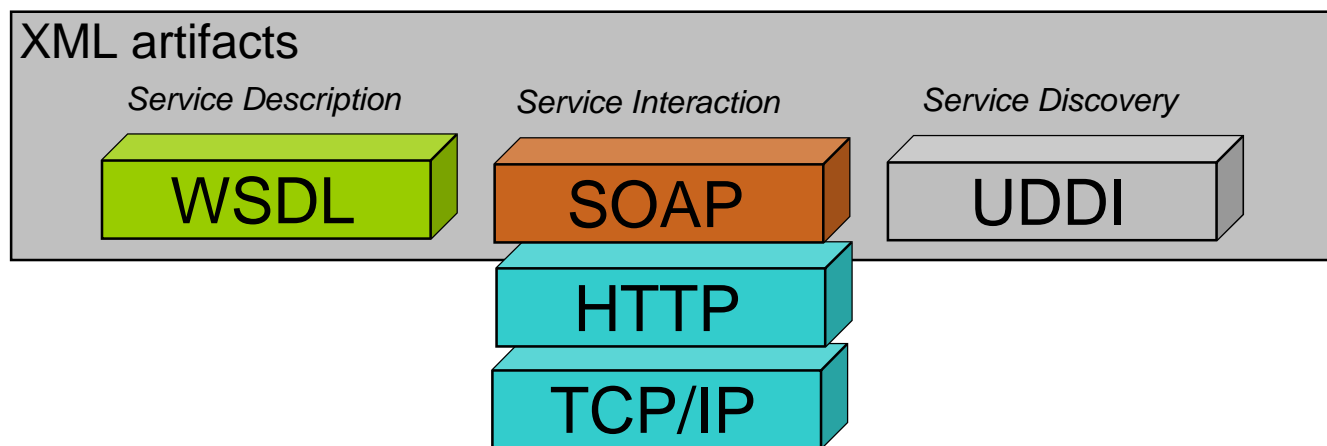
The Challenge

- Integrating WebApps with DDS requires dedicated bridges for the AppServer platform



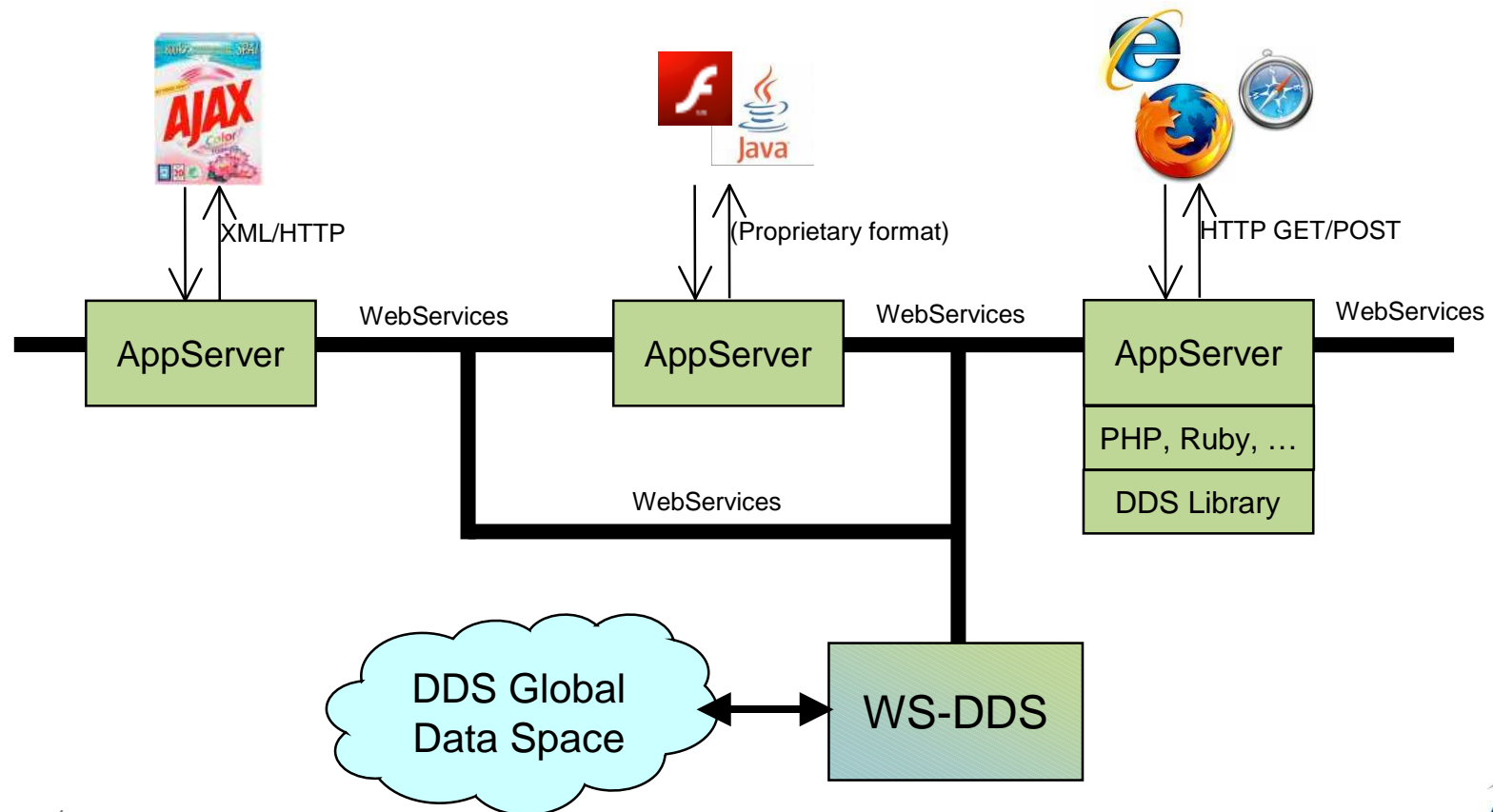
SOA-Web Services

- From W3C:
- *“A Web Service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered as XML artifacts. A Web Service supports direct interaction with other software agents using XML-based messages exchanged via Internet-based protocols”*

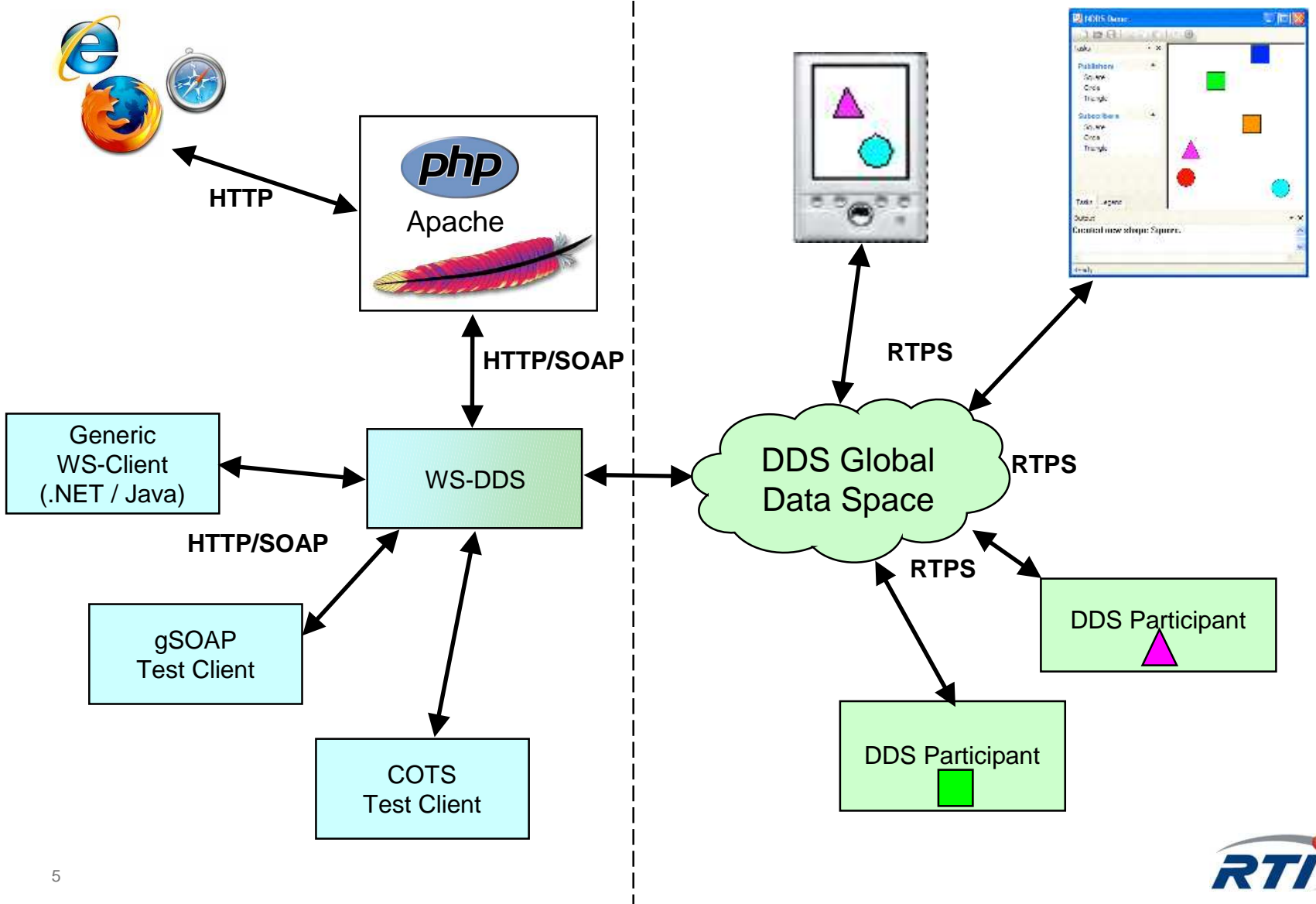


Solution

- DDS API exposed as WebService
- No need for dedicated bridge for AppServers



Demo



How can we make this real?

- Authentication / Access control
- Define DDS Operations:
 - Entity Creation?
 - Publication / Subscription
 - QoS configuration...
- Define Data [asynchronous] delivery mechanism
- Define Data Representation

Authentication & Access control

- Authentication:
 - **IP Address** and/or **user name & password** are provided for authentication
 - If login is successful, a **session token** is produced with a **time-to-live** that is updated on every WS-DDS call.
 - Token is used in all the **subsequent calls**.
- Access control:
 - Individual WS-clients can have access to different subsets of the WS-DDS API: read operations, write operations, notify operations.

DDS Operations

- WS-DDS does not include API for creating DDS DomainParticipants, Publishers, Subscribers and Topics.
- For simplicity these Entities are created automatically the first time you call **CreateDataReader** or **CreateDataWriter**.
- Since the interactions among Web Services are stateless, WS-DDS Server keeps the state of the DDS environment. Entities persist across sessions.
- QoS policies can be set by means of **QoSProfiles**.

Data Representation

- Data is delivered and received in a type-neutral form through JSON or XML
- Marshaling & Un-marshaling is performed by the WS-DDS transparently.

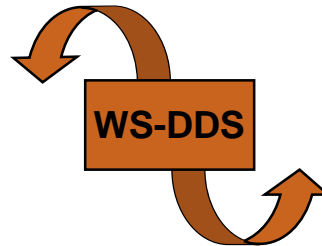
Type definition

```
struct Account {  
    long id;  
    string name;  
    long pin;  
    double balance;  
};
```

//@key

Sample (CDR)

143321
John Doe
1234
1345.90



JSON

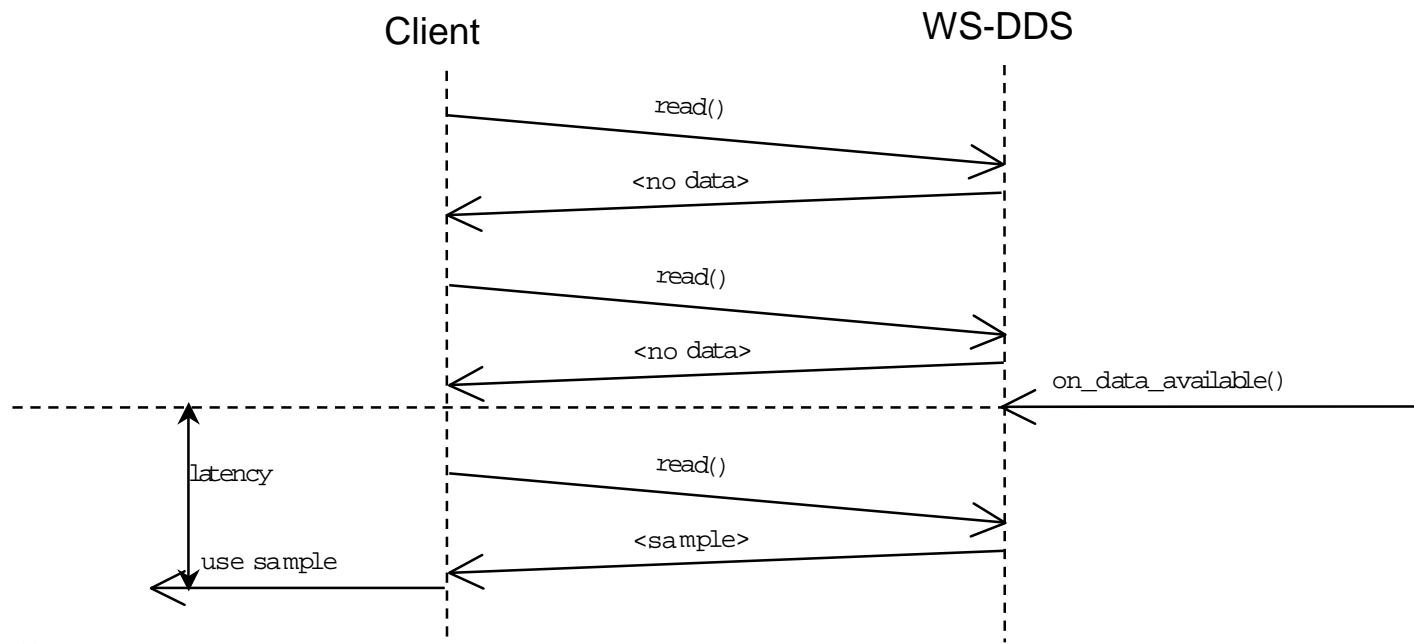
```
{  
    #id: 143321,  
    name="John Doe",  
    pin: 1234,  
    balance: 1345.90  
}
```

XML

```
<sample>  
  <id key='true' type='long'>143321</id>  
  <name type='string'>John Doe</name>  
  <pin type='long'>1234</pin>  
  <balance type='double'>1345.90</double>  
</sample>
```

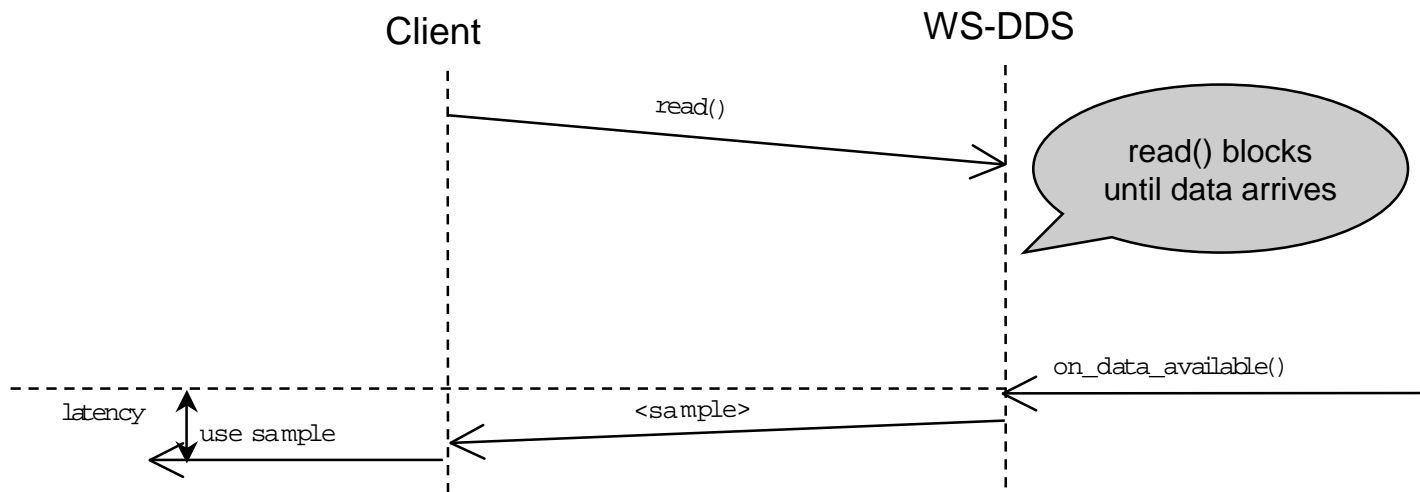
Data delivery: client polling

- Client periodically call read() to get the available samples
- All the received samples are then returned.
- Read() is not blocking.



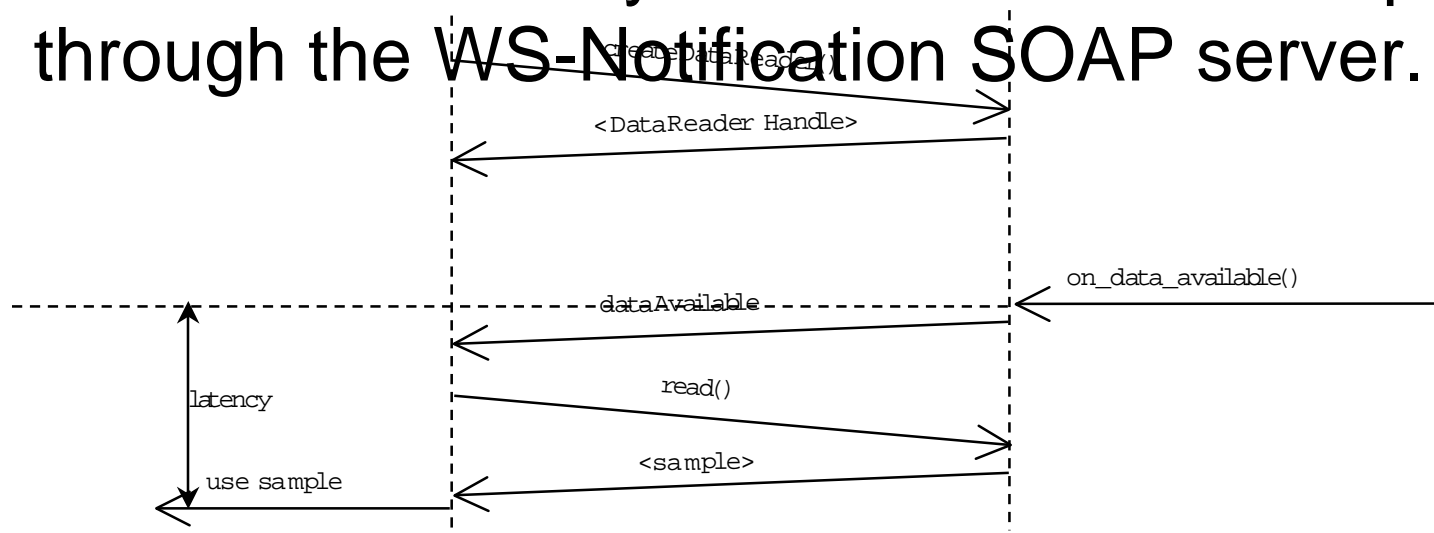
Data delivery: server push

- Client call read()
- If no samples are available, the calls block without sending back the HTTP response.
- When samples are received, WS-DDS completes the request



Data delivery: WS-Notification

- The client implements a SOAP server implementing WS-Notification.
- When Data data reader is created, the client passes the address of the WS-Notification server to WS-DDS.
- WS-DDS will notify the client of new samples through the WS-Notification SOAP server.

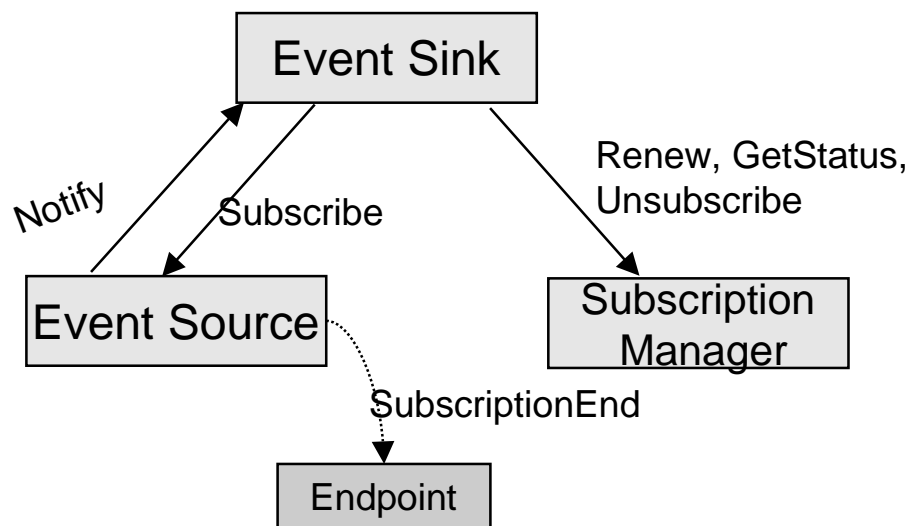


WS-DDS Eventing

- A Web Services wants to be notified when events occur in other Services and Applications
- Web Services involved
 - Subscriber Event Sink
 - Event Source
 - [Subscription Manager]
- Delivery mode
 - Push (only supported)
 - Pull

WS-Eventing Subscription manager

- Subscription manager is optional
- Event Source may define to be itself the Subscription manager or designate another Web Services



All together: the WS-DDS interface

- Authentication (WS-DDS Server)
 - Login
 - Logout
- DDS Operations (WS-DDS Server)
 - CreateDataReader
 - CreateDataWriter
 - Read
 - Write
- Notification (WS-DDS Client)
 - Notify

WS-DDS API (Authentication)

Operation	Request Message parameters	Response Message parameters
Login	string user_name string password	unsigned int return_code long session_id
Logout	long session_id	unsigned int return_code

WS-DDS API (DDS operations)

Operation	Request Message parameters	Response Message parameters
CreateDataReader	long session_id string topic_name string type_name int domain_id string qos_profile_name string http_address boolean push_style string content_filtered_topic	unsigned int return_code unsigned int reader_handle
CreateDataWriter	long session_id string topic_name string type_name int domain_id string qos_profile_name	unsigned int return_code unsigned int writer_handle
Read	long session_id unsigned int reader_handle int take_data long timeout int encoding	unsigned int return_code DataArray data
Write	long session_id unsigned int writer_handle string sample	unsigned int return_code

Access Control system

Users Table

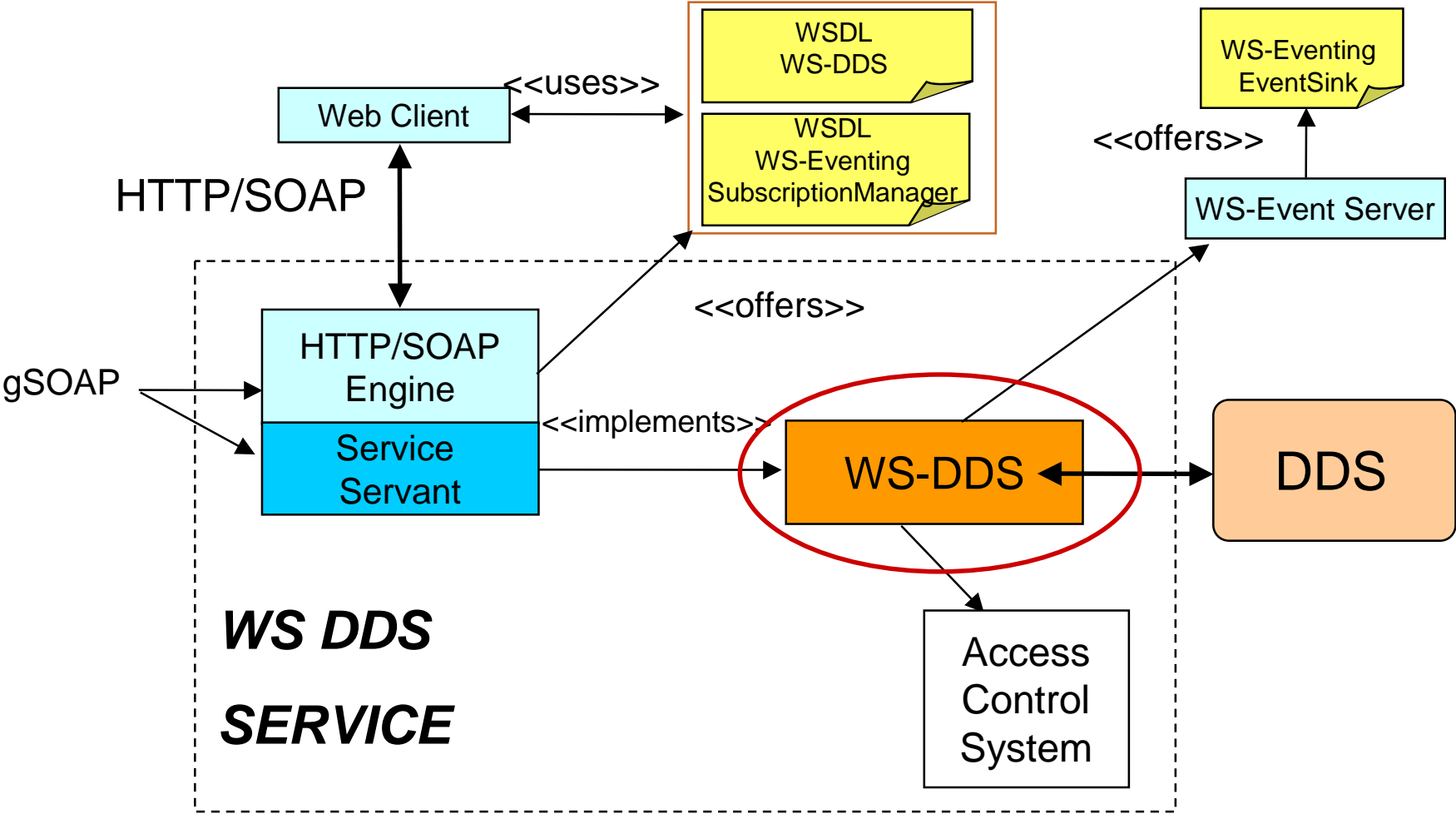
User Name (key)	Password	Roles
andrea	and_rti	admin, engineer, staff
fabrizio	fab_rti	engineer
gerardo	ger_rti	user

Topic_domain table

Domain (key)	Topic (key)	Read rights	Write rights	Notify rights
65	Square	staff, admin, user	staff, engineer, user	staff
65	Circle	admin, staff	*	staff, user
65	Triangle	admin, engineer	admin, engineer	staff, user

The information above are stored in two tables. The lookup in this files occurs only in the Login (users file) and CreateDataReader/Writer operations.

System components



WS-DDS Architecture (gSOAP)

- Blocks on listen socket
- Creates one thread per connection.
 - Dedicated to serve requests on the connection
 - Supports the blocking read
- Maintains tables with sessions and entities associated with each user/session
- A user can only have one open session
- Verifies access control on each operation

WS-DDS Performance

- **Throughput** (msgs per second) using gSOAP, and using a COTS version.
- Two different **connection types**: persistent and non-persistent connection (keep-alive)
- **Batching** reads (50 samples per operation) vs. single sample read.
- **Data representation**: XML / JSON
- All the tests performed **do not** include DDS data processing

Performance results (1)

- gSOAP / COTS
- Persistent connection
- 1 sample per read operation
- Format: JSON
- Read/Write operations

Operation	gSOAP (msg/sec)	COTS (msgs/sec)
Read	2055	918
Write	2055	918

gSOAP performs almost 2 times better than the COTS framework

Performance results (2)

- Format: JSON
- 50 samples per read operation
- Read operations

Framework	Persistent	Throughput (op/sec)	Throughput (msg/sec)
gSOAP	Yes	574	28705
gSOAP	No	406	20306
COTS	Yes	188	9.218

Persistent connection improves performance of about 40% (w/batching)

Performance results (3)

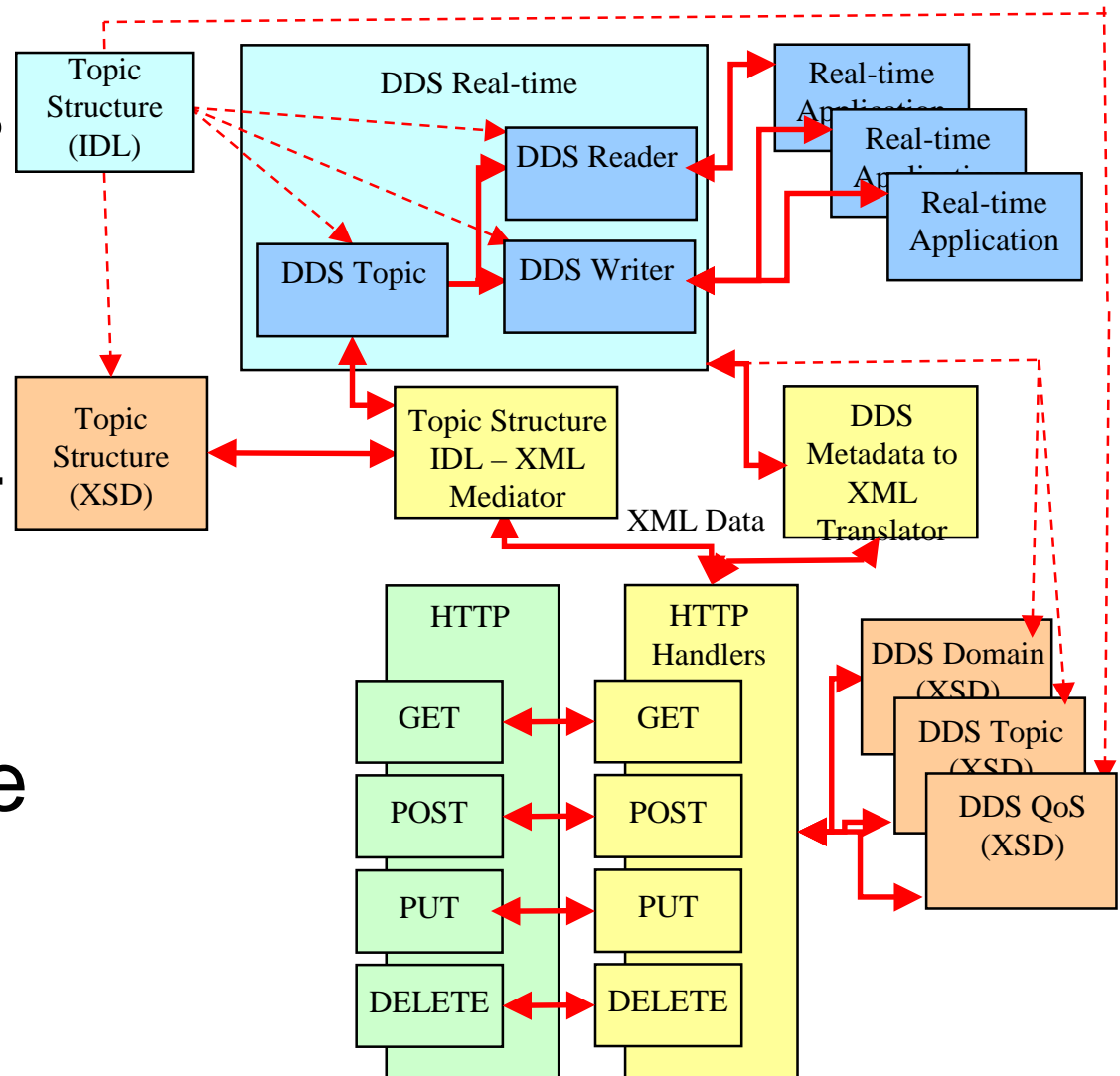
- Framework: gSOAP only
- Persistent connection

Operation	Encoding	Throughput (op/sec)	Throughput (msg/sec)
Single Read	XML	2,070	2,070
Single Read	JSON	2,054	2,054
50 sample read	XML	514	25,709
50 sample read	JSON	517	28,705
Write	XML	2,099	2,099
Write	JSON	2,135	2,135

JSON encoding can improve performance of only 12% (batching)

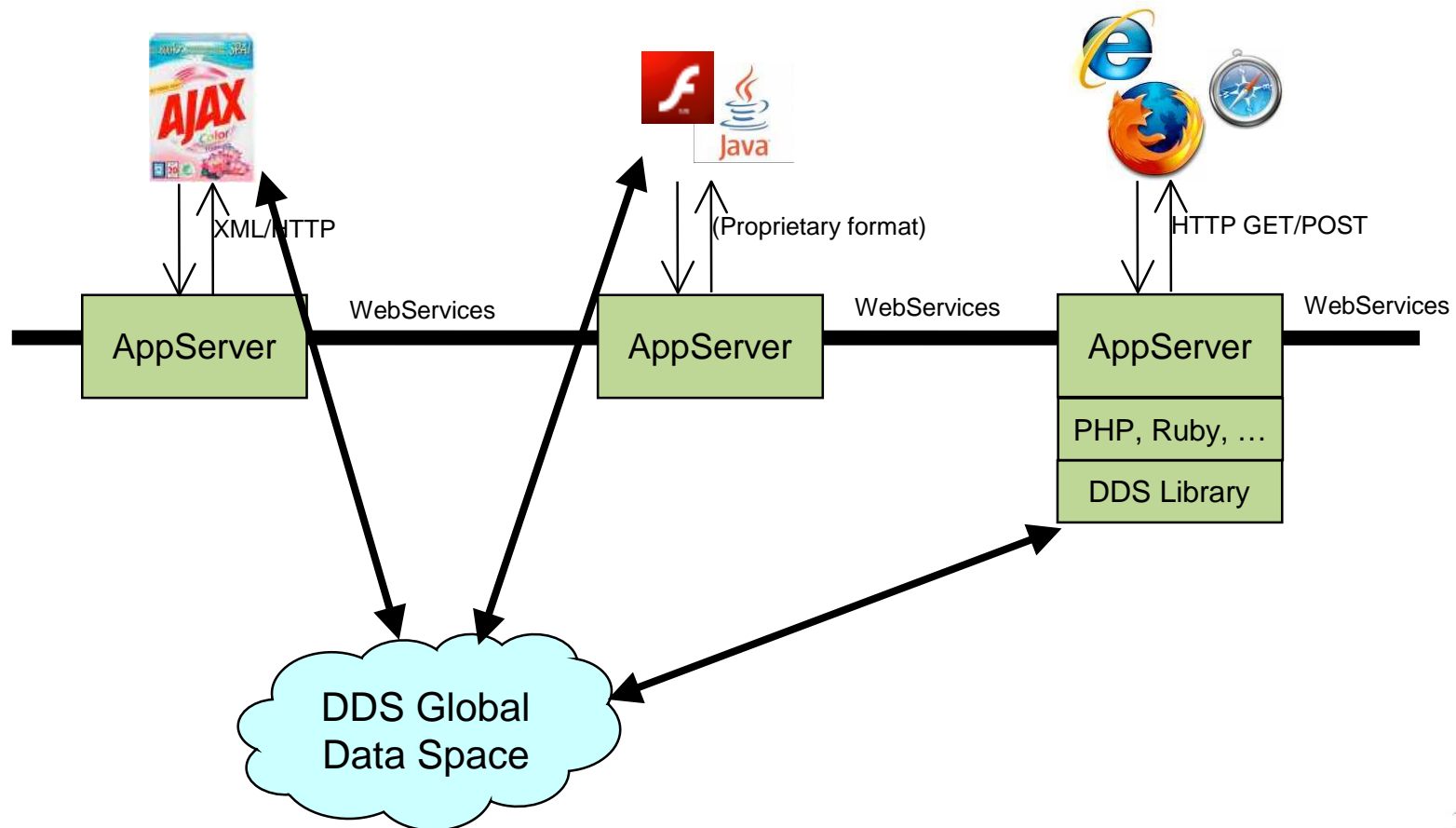
Use run time mediators

1. Standardize XML Schemas for Domains, Topics Names and QoS
2. Generate XML documents schemas that describe DDS Topic Structure
3. Create generalized mediators that use XML Data



The Challenge

- Integrating WebApps with DDS requires dedicated bridges for the AppServer platform



Example

- Open Geospatial Consortium
 - WPS - Web Processing Service: Remote processing service
 - *WPS makes it possible to publish, find, and bind to processes in a standardized and thus interoperable fashion. Theoretically it is transport/platform neutral (like SOAP), but in practice it has only been specified for HTTP. It is best described as a non-REST-ful RPC type service although it does comply with most of the REST principles.*

Conclusions

- WS-DDS provides an effective way to access DDS data from a web client
- The WS-DDS interface is very simple yet provides almost complete functionality
 - QoS profiles are a key feature
- Separating HTTP from WebServices increases the opportunities to use DDS
- Exposing DDS meta-data through XML schemas
- Supporting XML data itself as part of the standard

Other things to look at

- Comprehensive performance/scalability tests
- Sequence API for writes
- Analyze performance of DynamicType
- Delete API for entities
 - What to do with orphan entities
 - Garbage Collection ?
- Content filtered topic, time-based filter

Q&A



- Questions ?
- Preguntas ?
- Perguntas ?
- Domande ?
- Fragen ?
- Ερωτήσεις ?
- 質問 ?
- 問題 ?
- 질문 ?
- Vragen ?
- Вопросы ?