A RESTful Approach to the OGSA Basic Execution Service Specification

Sergio Andreozzi

Università di Bologna
sergio.andreozzi@unibo.it
http://sergioandreozzi.com/

Moreno Marzolla

INFN Padova
moreno.marzolla@pd.infn.it
http://www.dsi.unive.it/~marzolla

Talk outline

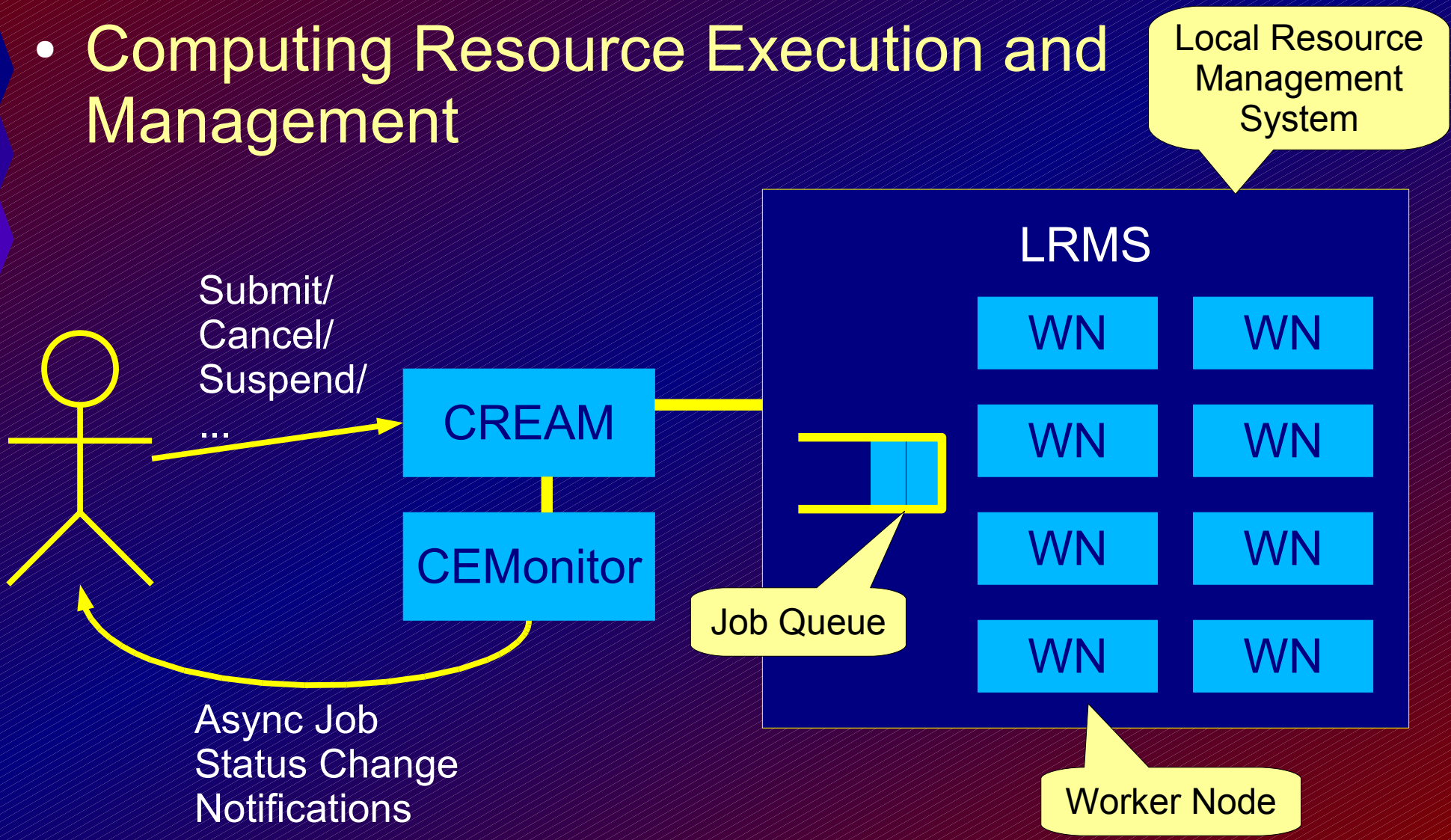
- Introduction
  - Job Management in Grid systems
  - The Basic Execution Service and Job Submission Description Language specifications
  - REST
- RESTful BES
- BES extensions
- Conclusions

Job Management in Grid Systems

- Job Submission is one of the basic functionalities of any Grid system
- Users submit and manage jobs (or *activities*) through an Execution Service
  - Start/Stop execution
  - Pause/Resume execution
  - Query job status

Example: the gLite CREAM service

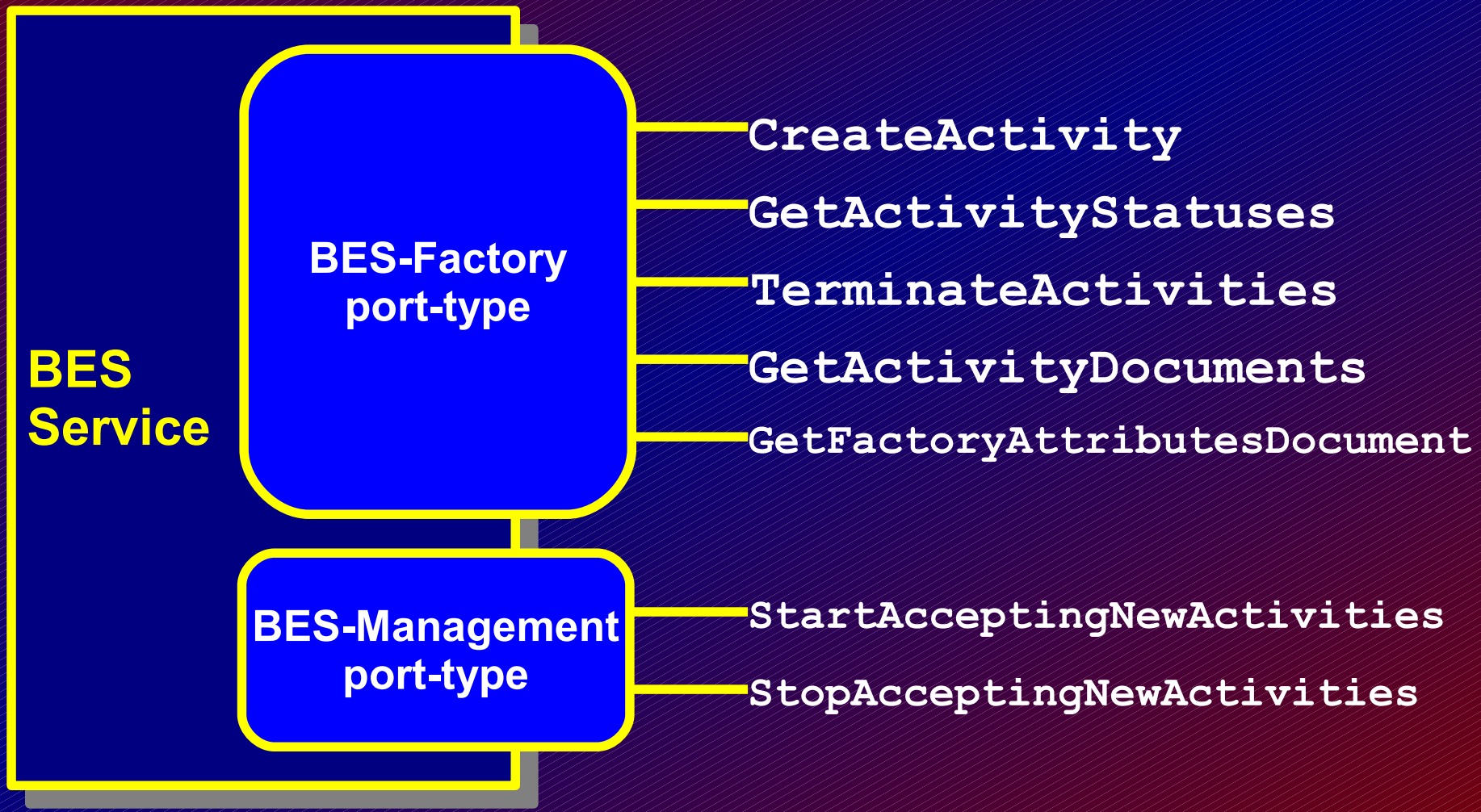
- Computing Resource Execution and Management



Achieving Interoperability with Open Standards

- Different Grids provide different interfaces for the job management service
  - Same functionalities, different names, datatypes...
- Job Submission Description Language (JSDL)
  - OGF Specification
  - XML-based notation for describing computational jobs and their resource requirements
- OGSA Basic Execution Service (BES)
  - OGSA Specification
  - WebService-based interface for job submission and management
  - Uses JSDL to describe activities

BES Port-types and operations



JSDL

```
<JobDefinition id="HelloWorldJob">
   <JobDescription>
      <JobIdentification>
         <JobName>Hello World</JobName>
         <Description>
            This job prints the Hello World message
         </Description>
         <JobProject>OMII-EU</JobProject>
         <JobAnnotation>uuid:090fe040e0</JobAnnotation>
      </JobIdentification>
      <Application>
         <ApplicationName>Echo</ApplicationName>
         <ApplicationVersion>1.0</ApplicationVersion>
         <Description>
            This application prints the Hello World
            message to the standard output
         </Description>
      </Application>
   </JobDescription>
</JobDefinition>
```

What is REST?

- REpresentational State Transfer
  - Fielding, R. T. and Taylor, R. N. *Principled design of the modern Web architecture*. ACM Trans. Internet Technol. 2, 2 (May. 2002), 115-150.
- It is an abstraction of the architectural elements within a distributed hypermedia system
- RESTful HTTP
  - XML + HTTP

RESTful HTTP data elements

- Resource
  - the conceptual target of a hypertext reference
- Resource identifier
  - URL, URN
- Representation
  - HTML document, JPEG image
- Representation metadata
  - media type, last-modified time
- Resource metadata
  - source link, alternates, vary
- Control data
  - if-modified-since, cache-control

Why RESTful HTTP?

- Considered somehow more simple and lightweight than WS
  - Uses standard HTTP methods
  - Reduces interoperability problems
- Many services offered both with WS and RESTful interfaces
  - Google GDATA API
  - Amazon S3
  - …

Building a RESTful service

1. Model your system as a set of resources
2. Give each resource a URI
3. Define the operations on the resources
   - Operations can be performed using standard HTTP operations (GET, POST, DELETE...)
4. Design the representation served to the client
   - The same information can be represented as plain text, XML, ...
5. Define the error conditions to be handled
   - HTTP error codes

RESTful BES

Identify Resources and Resource Identifiers

- **`/`**
  - representation of the service capabilities (BES factory attributes document)
- **`/status`**
  - current status of the BES service
- **`/activities/`**
  - the list of all activities present in the service submitted to the given share (e.g., batch queue)
- **`/activities/`*`id`***
  - the *current* representation of activity *id*

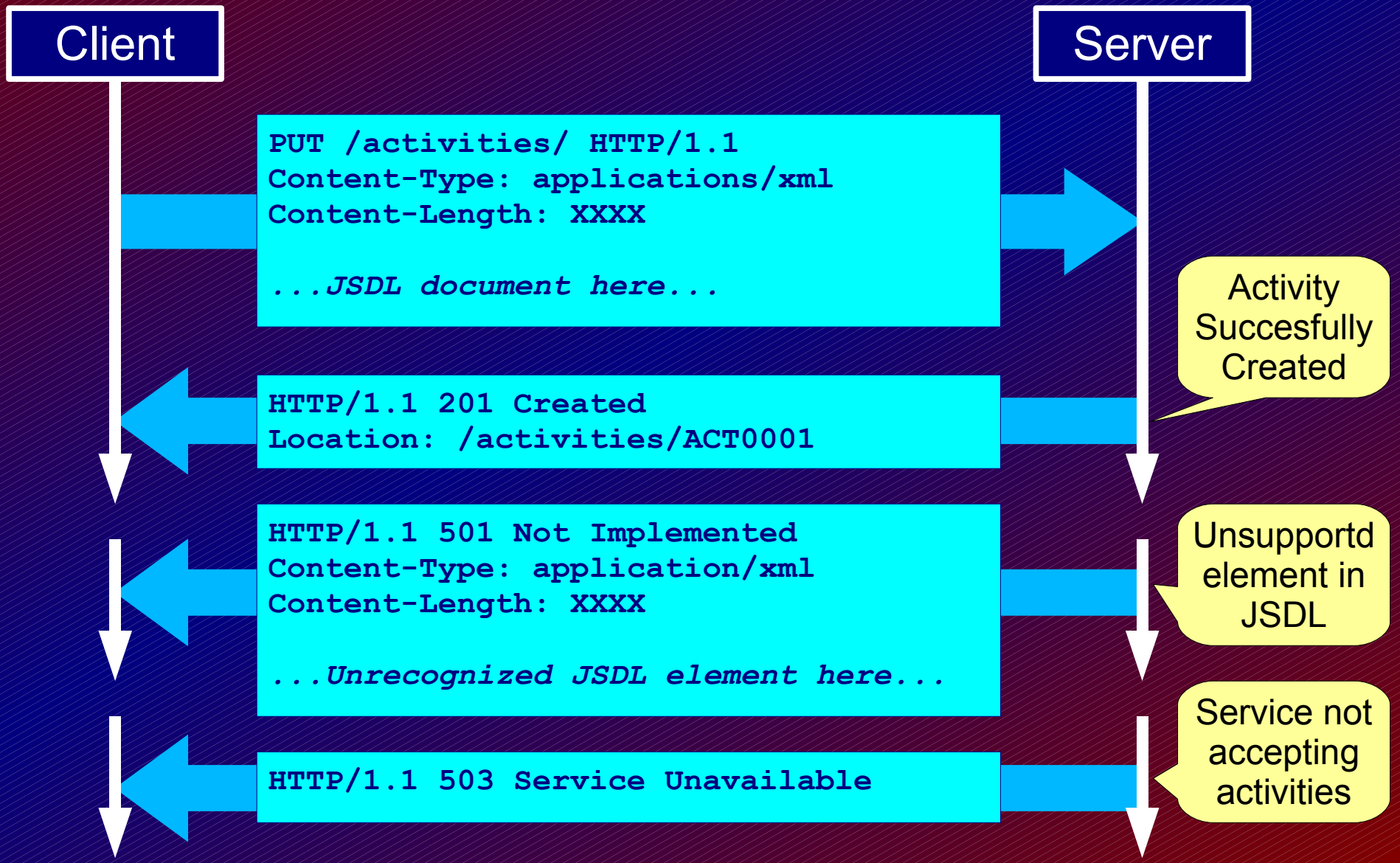
RESTful BES

Identify Resources and Resource Identifiers

- **`/activities/`*`id`*`/submitted`**
  - the JSDL document which has been used to instantiate activity *id*
  - In general, the execution service may return a different (processed) JSDL as the “current” JSDL
- **`/activities/`*`id`*`/status`**
  - the current status of activity *id*

RESTful BES / Operations

`CreateActivity`

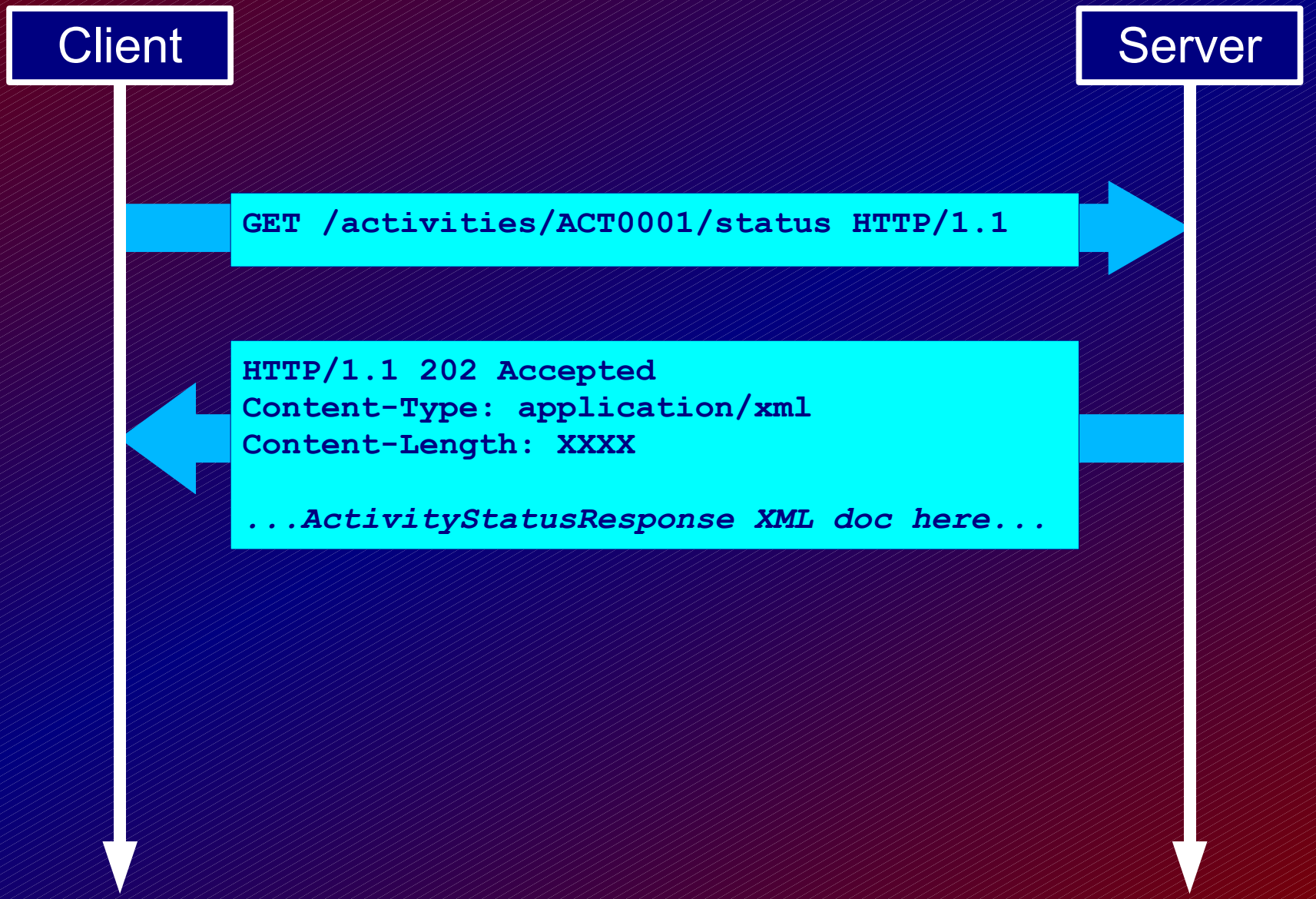


RESTful BES / Operations

GetActivityDocuments

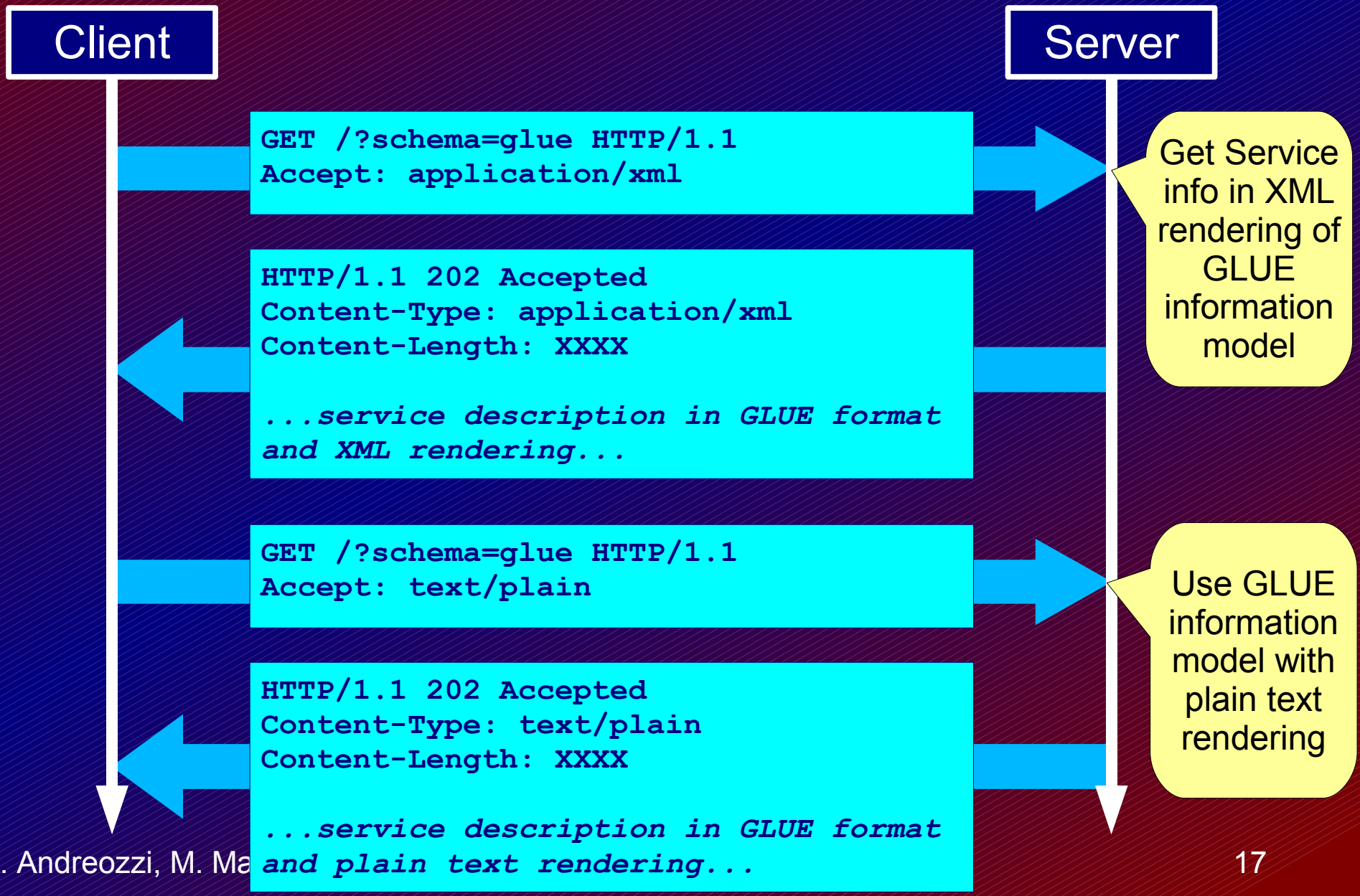
RESTful BES / Operations

GetActivityStatuses



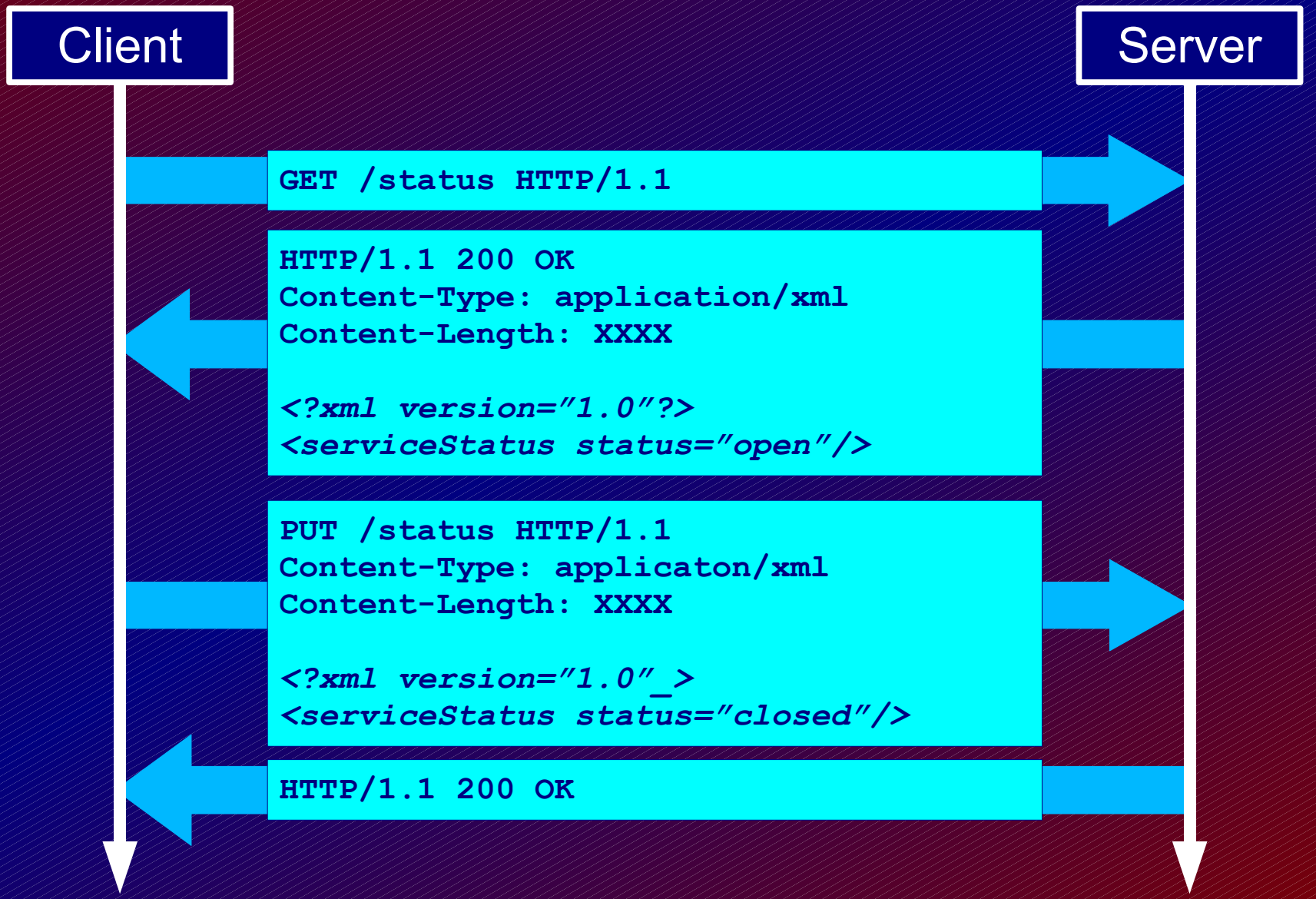
RESTful BES / Operations

GetFactoryAttributesDocument

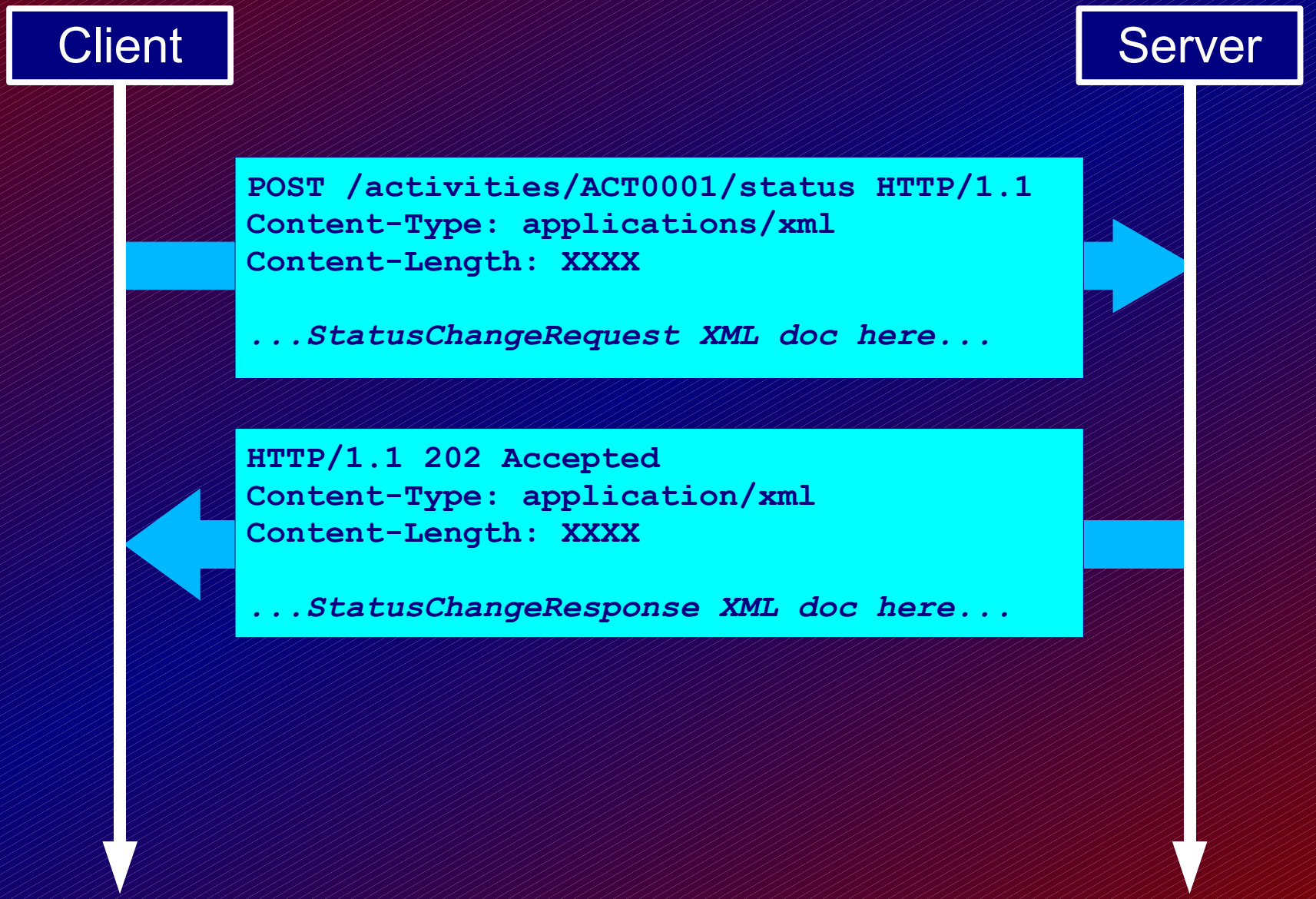


RESTful BES / Operations

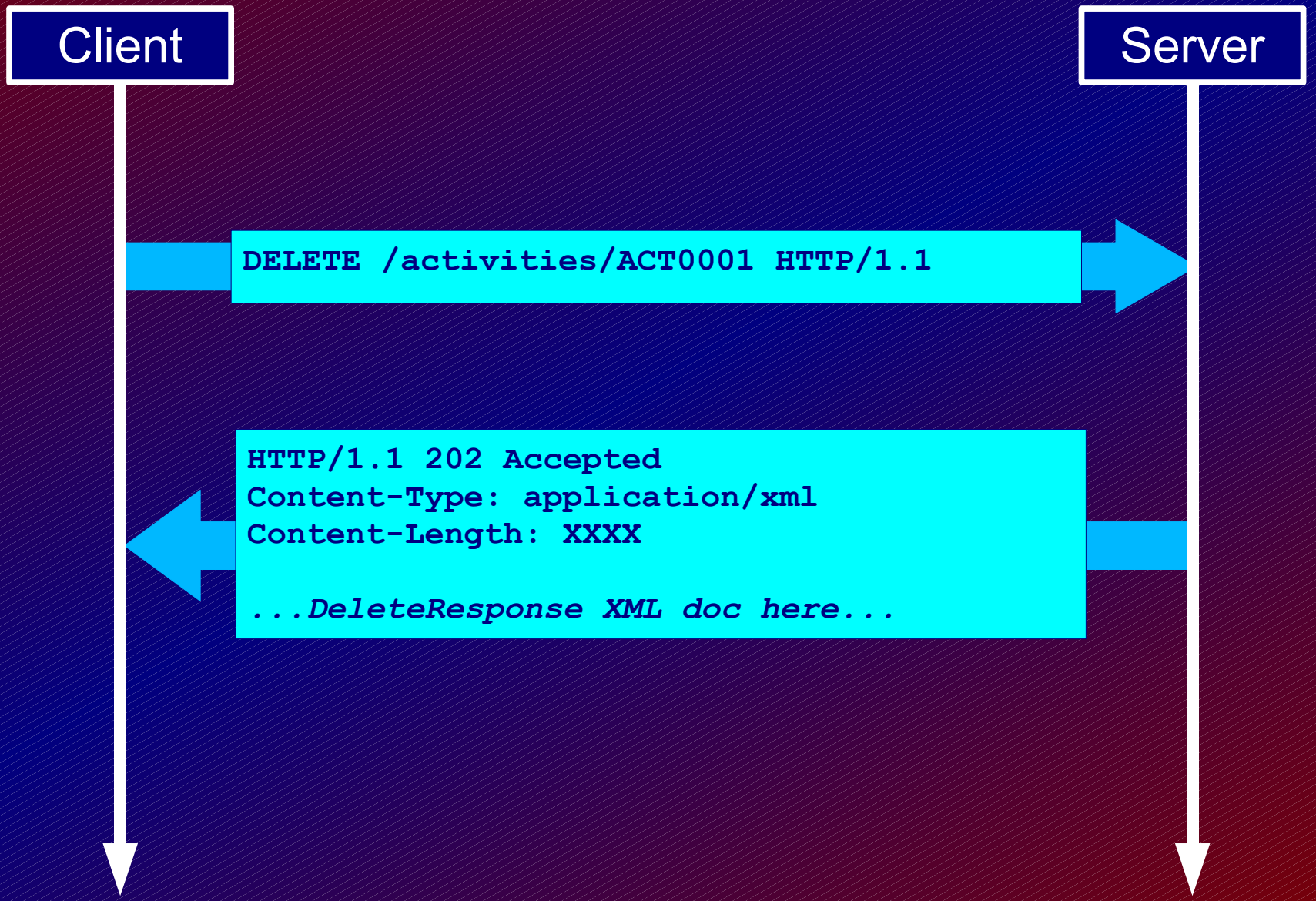
IsAcceptingNewActivities/SetAcceptingNewActivities



BES extensions
Change status of an activity



BES extensions
Delete (purge) an Activity



BES batch operations

- Some BES operations can operate on multiple activities
  - e.g., GetActivityStatuses(A_1, A_2, ... A_N)
- Encoding multiple activities ID in a single URI

```
GET /activities/ACT001;ACT002;ACT003/status HTTP/1.1
```

- Encoding multiple result codes in a single HTTP response

```
HTTP/1.1 202 Accepted
Content-type: application/xml
Content-length: XXXX

XML result data structure
```

BES extensions

- Idempotent execution
  - Makes CreateActivity idempotent

```
POST /activities/ HTTP/1.1
Pragma: IdempotentActivityID=ABCD0100
...
```

Client-generated nonce

- Lifetime Management
  - Allows clients to specify the maximum lifetime for an activity

```
POST /activities/ HTTP/1.1
Pragma: InitialTerminationTime=20090524
...
```

Conclusions

- The BES specification has been defined in term of WS technologies
  - WS are not the only way of implementing the Service Oriented Architecture (SOA) paradigm
- We showed how BES can be implemented as RESTful HTTP
  - ✔ Reduces complexity
  - ✔ BES operations naturally map to HTTP operations
  - ✔ BES extensions can be supported as well
  - ✗ Not possible to generate interfaces from WSDL
  - ✗ Difficult to encode vector operations
    - Identify multiple activities with a single URI
    - Multiple HTTP return status codes

Conclusions

Thank you!