AFS/Web

AFS file access through the web, a good idea ?

A. Mancini[1] M. Masi[2]

[1]Dipartmento di Matematica “U.Dini”
Università degli Studi di Firenze

[2]Dipartiemento di Sistemi e Informatica
Università degli Studi di Firenze

European AFS meeting 2009 Roma, September 28-30

What & Why

Authenticated access to the filesystem through the “web”

This is going to become a major request of users:
a web interface usable from everywhere, even a InternetCafè

Why a new project

- authentication at application level
- a rich-web-interface
- something we can integrate with existing/in development web interfaces
- It is FUN :)

What & Why

Authenticated access to the filesystem through the “web”

This is going to become a major request of users:
a web interface usable from everywhere, even a InternetCafè

Why a new project

- authentication at application level
- a rich-web-interface
- something we can integrate with existing/in development web interfaces
- It is FUN :)

What & Why

Authenticated access to the filesystem through the “web”

This is going to become a major request of users:
a web interface usable from everywhere, even a InternetCafè

Why a new project

- authentication at application level
- a rich-web-interface
- something we can integrate with existing/in development web interfaces
- It is FUN :)

... some considerations

Web Browsers are just EVIL

- not two browsers have similar javascript interpreters
- POST/GET Urlencoding/Multipart Syncronous/AJAX ...

- use a high level library: GWT
- do not rely on the web interface, finally it is just a GUI

How

1. delegate authentication to a dedicated service (Identity Provider)
2. provide access to the filesystem only through specific services (FileProviders)
3. adopt a standard (SAML in our case) for exchanging security assertions

... some considerations

Web Browsers are just EVIL

- not two browsers have similar javascript interpreters
- POST/GET Urlencoding/Multipart Syncronous/AJAX ...

- use a high level library: GWT
- do not rely on the web interface, finally it is just a GUI

How

1. delegate authentication to a dedicated service (Identity Provider)
2. provide access to the filesystem only through specific services (FileProviders)
3. adopt a standard (SAML in our case) for exchanging security assertions

... some considerations

Web Browsers are just EVIL

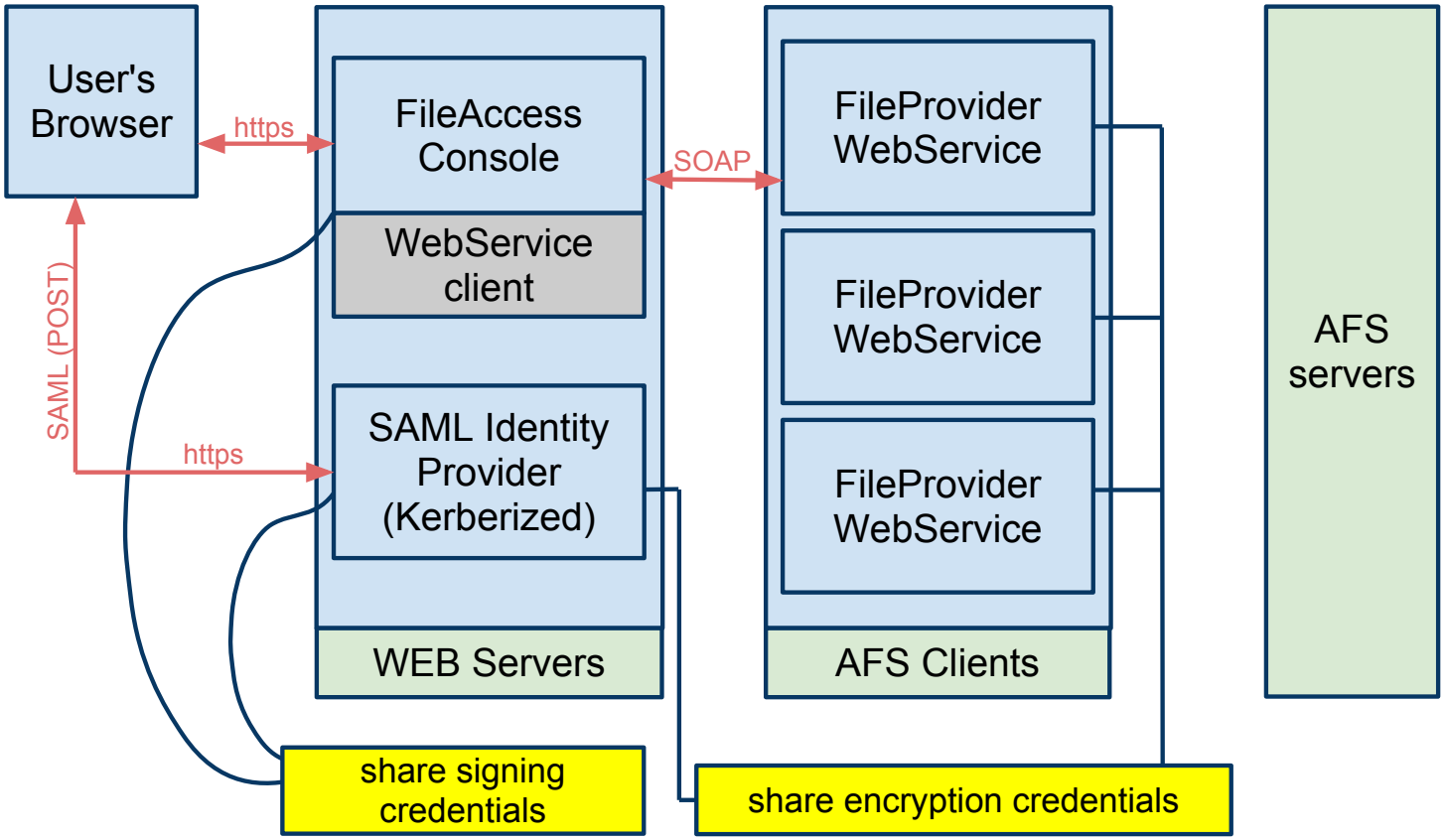
- not two browsers have similar javascript interpreters
- POST/GET Urlencoding/Multipart Syncronous/AJAX ...

- use a high level library: GWT
- do not rely on the web interface, finally it is just a GUI

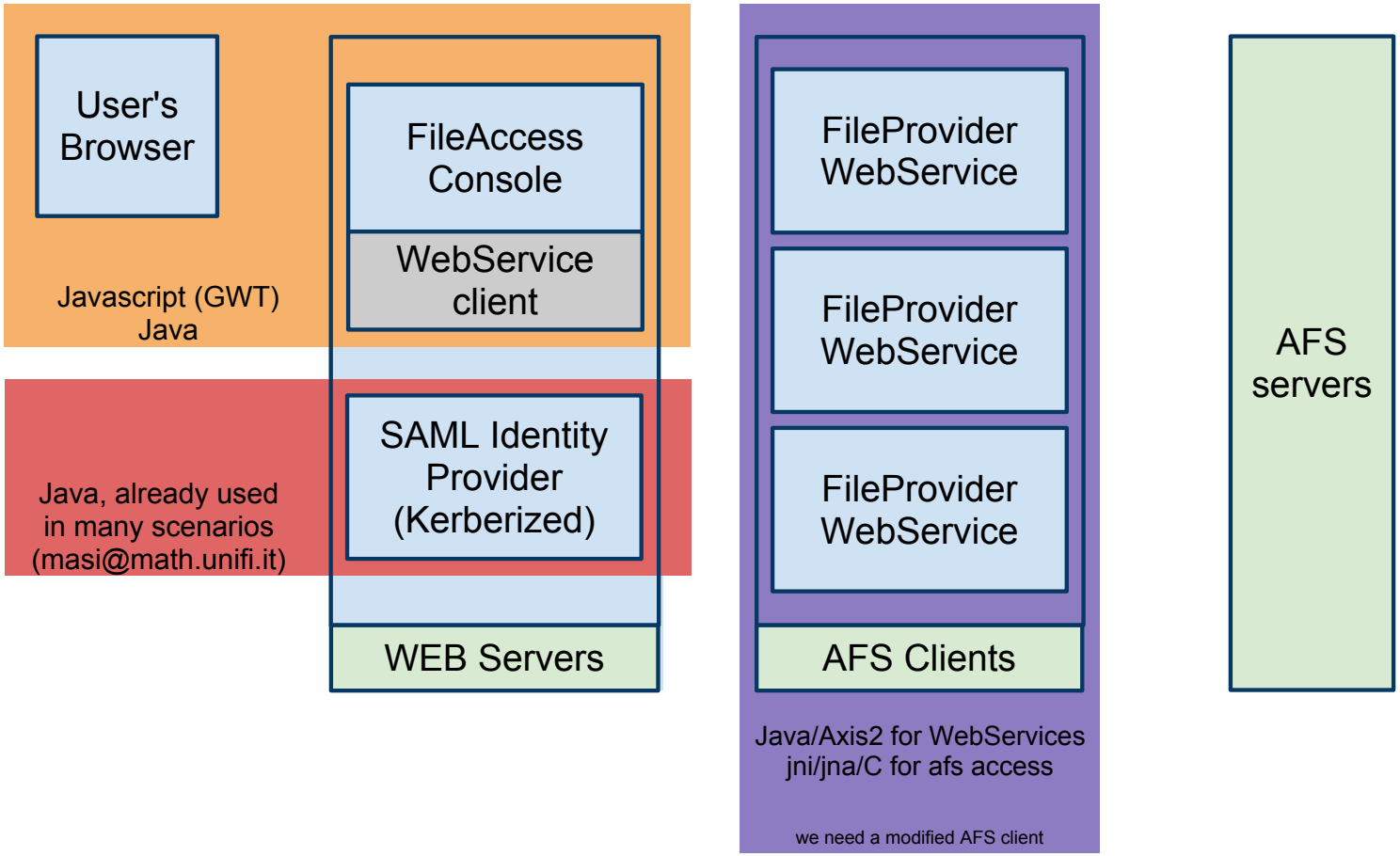
How

1. delegate authentication to a dedicated service (Identity Provider)
2. provide access to the filesystem only through specific services (FileProviders)
3. adopt a standard (SAML in our case) for exchanging security assertions

The big picture



The big picture: code (to be) written



FileProvider

Very simple object: (almost) standard OpenAFS clients.

1. receive a request
   1. “enter” in a PAG
   2. authenticate
   3. operate on file-system
   4. trash authenticated state
2. stream response

for definitions enthusiasts:

Fileproviders expose the filesystem as WebService.

Impl’d with apache’s Axis2.

stateless

FileProvider

Very simple object: (almost) standard OpenAFS clients.

1. receive a request
   1. “enter” in a PAG
   2. authenticate
   3. operate on file-system
   4. trash authenticated state
2. stream response

for definitions enthusiasts:

Fileproviders expose the filesystem as WebService.

Impl’d with apache’s Axis2.

stateless

FileProvider

Stateless WebService for a Price

Stateless

have to reauthenticate on each run

WS communicate through XML

- must use SOAP MTOM (overhead in binary data delivery)
- have to provide a format (schema) for messages

```
<fpns:getDirectoryListing>
    <fpns:path>/math.unifi.it/.../afsbp</fpns:path>
</fons:getDirectoryListing>

<fpns:directoryListing>
    <fpns:directory>Tutorial</fpns:directory>
    <fpns:directory>Notes</fpns:directory>
    ...
    <fpns:file fpns:mime="...">slides.eps</fpns:file>
</fpns:getFile>
```

Gain ? JSON may be an option using Axis2

FileProvider

Very simple object: (almost) standard OpenAFS clients.

1. receive a request
   1. “enter” in a PAG
   2. authenticate
   3. operate on file-system
   4. trash authenticated state
2. stream response

for definitions enthusiasts:

Fileproviders expose the filesystem as WebService.

Impl’d with apache’s Axis2.

stateless

FileProvider
PAG/Authenticate

... run in an application server (tomcat, jboss) and thread pools are mandatory for performance (mostly impossible to do without) ...

associate (P)AG to threads

We have a working implementation for the linux kernel using keyrings, sent to openafs-devel for comments.

(else we may end up with the entire threads-pool sharing credentials)

JAFS

Not ready for Kerberos authentication, update available.

TODO:

Why not trying to use user-space-openafs-client ?

FileProvider

Very simple object: (almost) standard OpenAFS clients.

1. receive a request
   1. “enter” in a PAG
   2. authenticate
   3. operate on file-system
   4. trash authenticated state
2. stream response

for definitions enthusiasts:

Fileproviders expose the filesystem as WebService.

Impl’d with apache’s Axis2.

stateless

FileProvider
Operating on the Filesystem

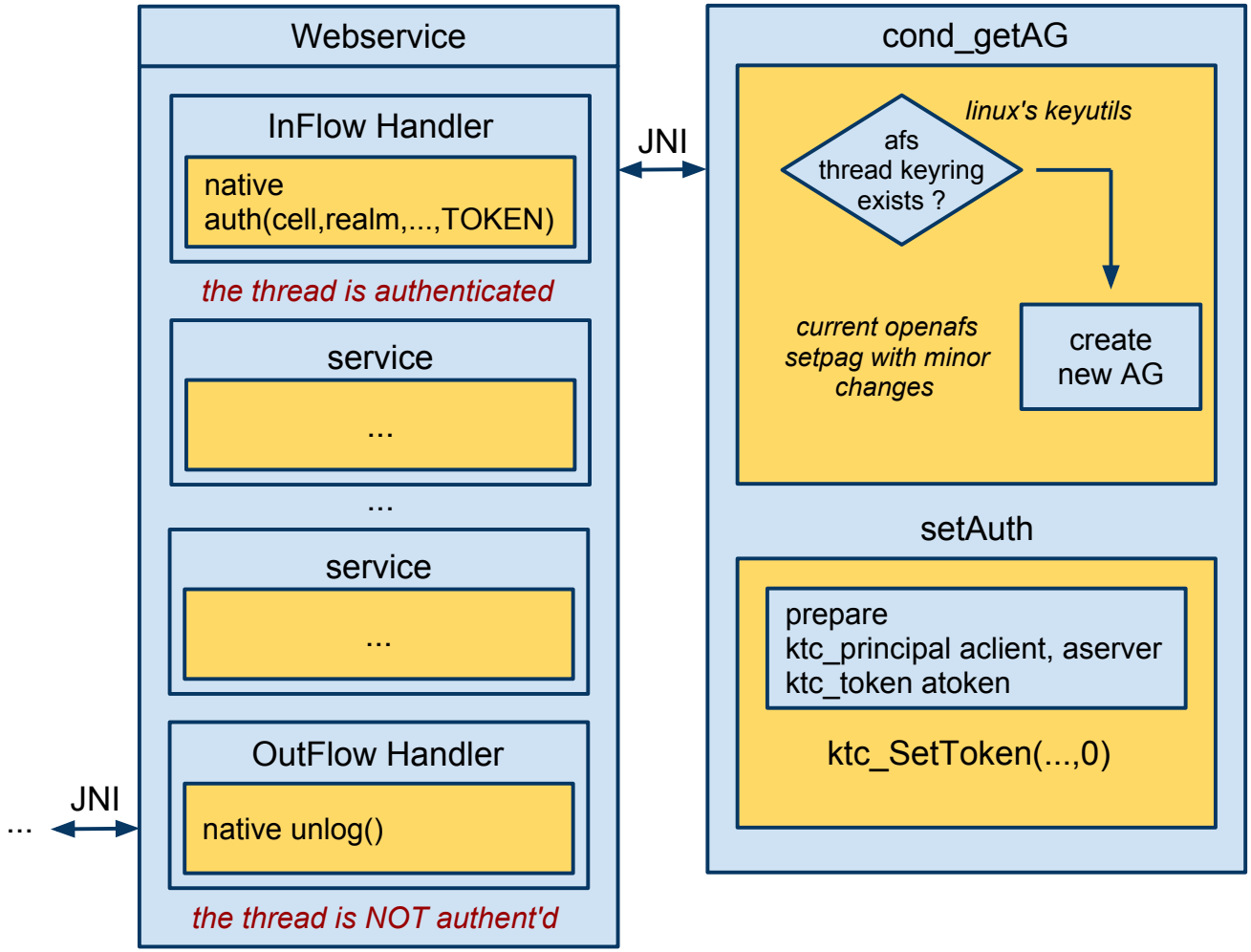
Java $\Longleftrightarrow$ afs

- accessing files does not require any special care (as long as we use in kernel cache mgr)
- accessing ACL's, FIDs, Volumes does require special treatment
- honestly, my feeling is that JAFS does its work (?)

Question:

Updating JAFS or rewriting (so we can support uafs too) ?

FileProvider



Token ?

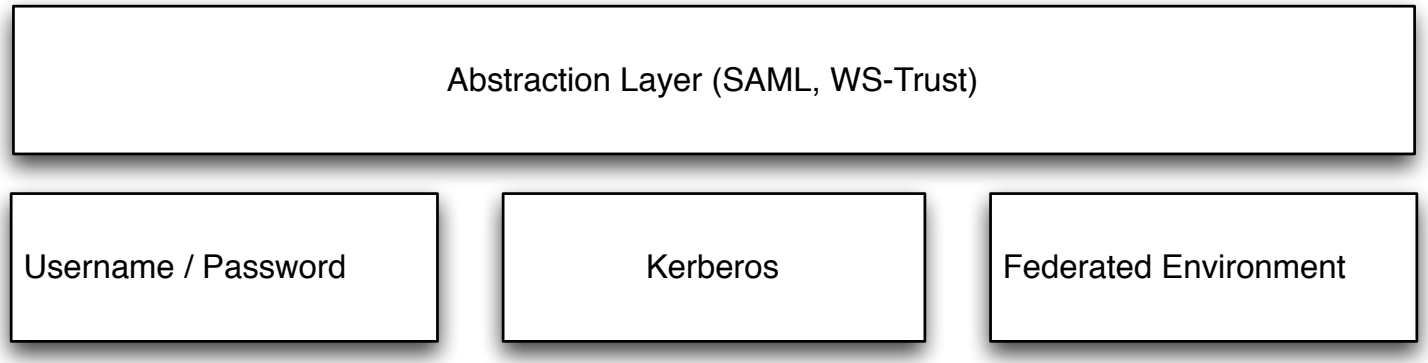
The Problem

Nowadays, we have dozens digital identities

- Username and password for the webmail account
- Username and password for the bank account
- Username and password for frequent flyer status
- A Kerberos account for the domain's workstation
- A key pair for signing documents
- . . .

The current solution: a new abstraction layer (?)

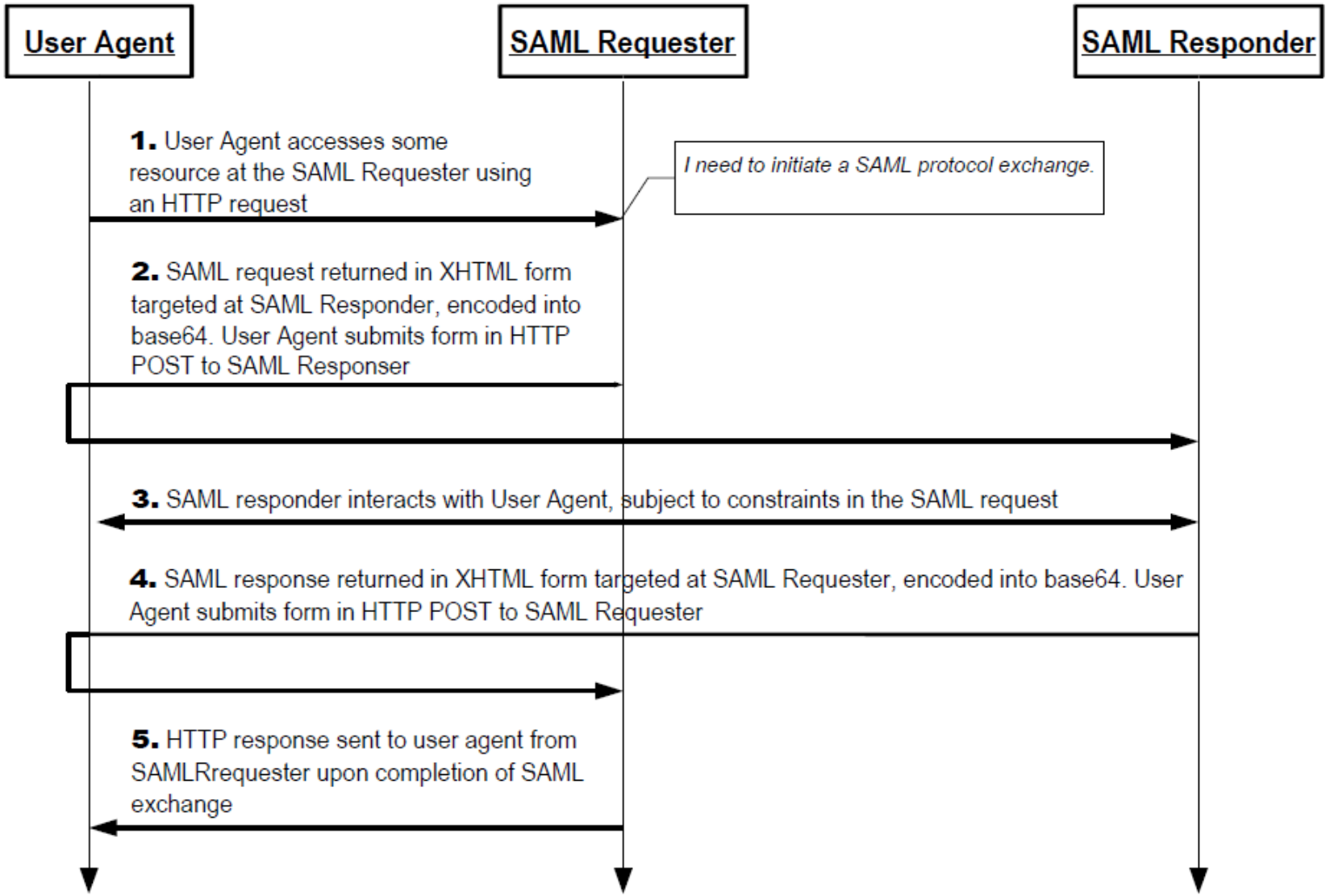
- Security Assertion Markup Language (SAML) (OASIS)
- SAML is a method for encoding security *assertions* (i.e. tokens) in XML, and a method (i.e. a protocol) to exchange such assertions.



The Identity Provider

- The Identity Provider is a service that *provide* a *trusted identity* to a set of nodes, by asserting an authentication made in the past by an underlying mechanism (such as kerberos)
- From 10.000 feets, SAML indicates that an authentication have been performed by a certain user. It writes it in an XML document, to be sent over the network, using SOAP or whatever else.
- The transport mechanism (and other implementation details) are written in SAML Profiles (for SOAP, HTTP, Web Browser)

Web Browser SSO Profile for SAML2



Our identity provider

- Once received the username and password, IdP authenticates user against the KDC:
  - String cachename = "krb5_cache_"+UUIDGenerator.getUUID();
  - KrbAcquireTGT krbClient = new KrbAcquireTGT();
  - Long stoptime = 2000L; //2 Seconds, anybody knows why?
- Then we get the TGS for the AFS service and we convert it into a token, then we represent the token as an XML document

```
<tns:AFSToken xmlns:tns="urn:it.unifi.math:AFS">
    <tns:Principal>...</tns:Principal>
    <tns:StartTime>1253699537</tns:StartTime>
    <tns:EndTime>1253735535</tns:EndTime>
    <tns:SessionKey>....</tns:SessionKey>
    <tns:Ticket>.........</tns:Ticket>
</tns:AFSToken>
```

- finally we encrypt the token using the public key of the service providers
- and pack in a signed SAML assertion

```
<saml:Assertion ID="7811e9814bf51bc24318030c593375af"
                IssueInstant="2009-09-23T09:57:51.902Z" Version="2.0">
  <saml:Issuer>urn:idp:math.unifi.it:identity-provider</saml:Issuer>
  <ds:Signature>...</ds:Signature>                          <!-- signature -->
  <saml:Subject>
    <saml:NameID>mancini</saml:NameID>
    <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
      ...                                                  <!-- user principal etc... -->
    </saml:Subject>
    <saml:Conditions NotBefore="2009-09-23T09:57:51.902Z"
                     NotOnOrAfter="2009-09-23T19:57:51.902Z"> <!-- Kerberos Ticket validity --
      <saml:AudienceRestriction>
        <saml:Audience>http:....</saml:Audience>           <!-- web application -->
      </saml:AudienceRestriction>
    </saml:Conditions>
    <saml:AuthnStatement AuthnInstant="2009-09-23T09:57:51.902Z"
        SessionNotOnOrAfter="2009-09-23T19:57:51.902Z">
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>
          urn:oasis:names:tc:SAML:2.0:ac:classes:kerberos
        </saml:AuthnContextClassRef>
      </saml:AuthnContext>
    </saml:AuthnStatement>
    <saml:AttributeStatement>
      <saml:EncryptedAttribute>...Token...</saml:EncryptedAttribute><!-- encrypted token -->
      <saml:Attribute FriendlyName="REALM" Name="urn:REALM" NameFormat="urn:REALM">
        <saml:AttributeValue xmlns:xs="http://ww...org/XMLSchema" xsi:type="xs:string">
          MATH.UNIFI.IT
        </saml:AttributeValue>
      </saml:Attribute>
    </saml:AttributeStatement>
  </saml:Assertion>
```

Web Interface (named fileconsole)

Having

- the IdentityProvider to provide SAML Assertion(s)
- the FileProvider(s) that does the actual work on the filesystem

the WebInterface and has just to:

1. manage SAML assertions (entierely in the browser right now)
   a sort of in browser crendentials cache
2. assist the user in creating SOAP requests and decoding responses (and doing actual requests)
3. handle the browser *(get multipart data for uploads, GWTRPC where possible, etc... repsond correct content-data on downloads etc ...)*

Some technicalities may worth a note (e.g. getting the credentials back to the GWT application after a successfull autentication) but there is not time now, and afterall it is not AFS stuff.

Current Status

1. Not ready for Production (eclipse says: 43 TODO's)
2. On test servers works
3. IdP works very well
4. FileProvider (No schema for communications, authentication/(P)AGging needs test/j(u)afs issue
5. FileConsole, primitive, probably to be redesigned, assertions vs session identifiers in browser memory

AFS file access through the web, a good idea ?

Probably, but it is a lot of work !

That's All !

How about a demo ? (https://nettunio.math.unifi.it/console)