RestFS: Resources and Services are Filesystems, Too

Joe Kaylor, Konstantin Läufer, George K. Thiruvathukal
Presentation Outline

- Motivation
- Web service composition
- Local machine software composition
- Inter Process Communication
- Bridging the Gap: RestFS
- The Filesystem as a Connector Layer
- Examples and exploration
Motivation

- Currently, web services can be accessed through HTTP clients and HTTP libraries over network sockets.
- While some web services are accessible over filesystem interfaces, there is no general support or uniform filesystem interface.
- RestFS's goal is to provide a general filesystem interface to manage configuration, authentication, and interaction with RESTful web services.
Service Composition

- Mashups / Remixes
- WS-BPEL - Web Services Business Process Execution Language
- CORBA - Common Object Request Broker Architecture
Local Software Composition

- Libraries
- Shared files
- Pipes
- Domain Sockets
- Shared memory pages / memory mapped files
Filesystems That Abstract a Web Service

- GmailFS - stores data in a GMail webmail account
- FlickrFS - abstracts the Flickr photo service as a filesystem (available on Ubuntu)
- davfs2 - a WebDAV filesystem
Types of Web Service Filesystems

- Application Filesystems - file systems that provide convenient interfaces for programs to interact. A good example is FlickrFS presenting image files
- Object or storage abstraction - Using a web service as a storage location for a filesystem. Two good examples are GMailFS and SSHFS
- Connector filesystems - File systems that abstract a method of IPC or a network protocol. Two good examples are Plan 9's netfs and RestFS
RestFS

- A connector filesystem that provides a filesystem abstraction for RESTful web services
- Requests to and responses from web services are made through filesystem system calls
- Connections are initiated through manipulation of configuration files in the filesystem
- OAuth authentication can be established by manipulating authentication files
RestFS - Configuration Files

- Configuration files specify the host, port, resource, and triggering FS call
- Triggering FS call is the system call that triggers the web service method

```xml
<?xml version="1.0" encoding="UTF-8"?>
<RestfulSetting>
  <FsMethod>utime</FsMethod>
  <WebMethod>GET</WebMethod>
  <FormName></FormName>
  <Resource>/1/users/show/joekaylor.json</Resource>
  <Host>api.twitter.com</Host>
  <Port>80</Port>
  <OAuthTokenPath>/auth/twitter</OAuthTokenPath>
</RestfulSetting>
```
RestFS - Resource Files

- Programs can read and write to resource files in a normal way.
- When the triggering filesystem system call is issued on the resource file, a HTTP method call will be issued.
- If the resource file contains data, it will be sent along with the call.
- The response will be written back to the resource file.
RestFS - Triggering Call Timeline

1. Triggering FS call
2. RestFS intercepts call
3. Connect to URL send request
4. Listen for requests
5. Accept request
6. Process request
7. Respond to request
8. Store response in resource file
9. Return from triggering FS call

Diagram Flow:
- Triggering FS call leads to RestFS intercepts call, which in turn leads to Connect to URL send request.
- Once the request is sent, the system listens for requests, accepts the request, processes it, responds, stores the response, and finally returns from the triggering FS call.
OAuth Authentication in RestFS

- A special '/auth' folder exists in every RestFS mount.
- When a user creates a folder in '/auth', 4 files are created:
  - `config` - contains Access Token, User Auth, and Request Token URLs
  - `status` - contains the current progress or errors encountered during authentication. If a manual step is required (such as a CAPTCHA test), this file contains the URL to visit
  - `verifier` - if a manual step is required, any PIN that needs to be associated with RestFS can be written to this file
  - `token` - after successful authentication, this file contains the authentication token
- The token file can then be referenced in configuration files
- The associated resource file will automatically work with OAuth
RestFS - Authentication Process

1. Create new folder in /auth

2. Complete 'config' file

3. Interaction required?
   - Yes: Visit site URL in 'status' file to obtain PIN
   - No: Token available in 'token' file

4. Write PIN to 'verifier' file
OAuth Files (config)

<?xml version="1.0" encoding="UTF-8"?>
<OAuthConfigFile>
    <Key>asdf3244dsf</Key>
    <AccessTokenURL>https://api.twitter.com/oauth/access_token</AccessTokenURL>
    <RequestTokenURL>https://api.twitter.com/oauth/request_token</RequestTokenURL>
    <Secret>147sdfkek</Secret>
</OAuthConfigFile>
OAuth Files (token)

```xml
<OAuthTokenFile>
  <AccessToken>2534534asdf2348</AccessToken>
  <RequestToken>aql2343</RequestToken>
  <TokenSecret>adfjdsI24522</TokenSecret>
</OAuthTokenFile>
```
Software Composition with RestFS

- With RestFS, it is possible to compose remote web services with local software.
- This software can then be recomposed into new RESTful web services or other local software compositions.
- Can allow many computers to participate in local software compositions of a RESTful service by mounting a RestFS volume as a network filesystem over Samba or NFS.
Demonstrations

- Google Search
- Yahoo! Place Finder
- Twitter