NativeSPClusteringByProxy

If you use Apache and all you need Shibboleth for is to create your application's own session you don't really need any host affinity at all. You can use proxying to assure that all login activity occurs on the same host.

Example application

This is only a guide. Your installation and experience will differ.

Assume an implementation like this:

- You have several hosts supporting an application.
    - App is: 'app.example.edu'
    - Hosts are: 'srv1.s.example.edu', 'srv2.s.example.edu', 'srv11.s.example.edu'.
- To initiate a session your app redirects users to, say, "/login/...", a Shib protected location, where you generate an application session (with cookie) and redirect users back to "/app/...".

So, the accesses during a login are:

1. (user needs login)
2. /login/...
3. (user dialog with an IdP)
4. /Shibboleth.sso/...
5. /login/...
6. (back to the app with a session)

For the shibboleth part of the login to work '4' and '5' have to be on the same host. Starting with about version 2.4 we cannot proxy '4'. Its verification of server name is not optional. So we have to proxy the return to the login ('5').

SP Configuration

This is a much simpler implementation than my original. I think it works better all around.

Enable apache rewrites

```
RewriteEngine on
RewriteLog /logs/rewrite.log
RewriteLoglevel 0
```

Disable address checking

The proxied requests will come from the peer host, not the browser.

```
<Sessions checkAddress="false" consistentAddress="false" ...>
```

Choose a login path

For this example we'll use "/login". You could also use "/secureloign", for example, for 2-factor logins. Note that we don't protect the 'logiin' path with shibboleth. Instead we protect a hidden path. Call it "/login-shib".

Choose an 'in-progress' cookie

For this example we'll use "`splogin`".

Notify peers when the local host processes the /Shibboleth.sso

Set the appropriate cookie. And clear it when its been used.

This for host srv1:

```
RewriteCond %{REQUEST_URI} /Shibboleth.sso
RewriteRule ^(.*)$  - [CO=splogin:srv1x:app.example.edu:1:/:secure]


RewriteCond %{REQUEST_URI} !/Shibboleth.sso
RewriteCOnd %{HTTP_COOKIE} splogin
RewriteRule ^(.*)$  - [CO=splogin:srv1x:app.example.edu:-1:/:secure]
```

Similar configuration for the other hosts.

Proxy logins started on peer hosts

Detect both by cookie and path. This again for host srv1.

We need to detect the login path and proxy to the host that processed the /Shibboleth.sso.

```
# login session on srv2
RewriteCond %{REQUEST_URI} /login
RewriteCond %{HTTP_COOKIE} splogin=srv2x
RewriteRule ^/(.*)$  https://srv2.s.example.edu/$1 [P]

# login session on srv11
RewriteCond %{REQUEST_URI} /login
RewriteCond %{HTTP_COOKIE} splogin=srv11x
RewriteRule ^/(.*)$  https://srv11.s.example.edu/$1 [P]
```

Place similar configurations on the other hosts.

Rewrite unproxied logins to the protected path

This is to catch the initial redirect to login.

```
RewriteCond %{REQUEST_URI} /login
RewriteCond %{HTTP_COOKIE} !splogin
RewriteRule ^/login/(.*)$  /login-shib/$1 [PT]
```

Protect the real shib login path

Any require lines should be OK.

```
<LocationMatch /login-shib>
AuthType shibboleth
require valid-user
order allow,deny
allow from all
</LocationMatch>
```

Let shibboleth silently handle the other host login paths.

When handling the redirect from /Shibboleth.sso we need to gather the shib attributes. This has to be just "require shibboleth".

```
<LocationMatch /login>
AuthType shibboleth
require shibboleth
order allow,deny
allow from all
</LocationMatch>
```

Testing

I use webisoget to test the proxy setup. Use its '-maxhop 1' option to single step through the many login redirections. You have to use your /etc/hosts file to direct the requests to particular hosts. The '-map' option won't work because libcurl caches mapped dns addresses and there's no way to prevent that (short of editing libcurl). The /etc/hosts file works well, though. You may want to increase the lifetime of your cookies during testing.

That's it!

I think.