ProtoGENI Component Manager API

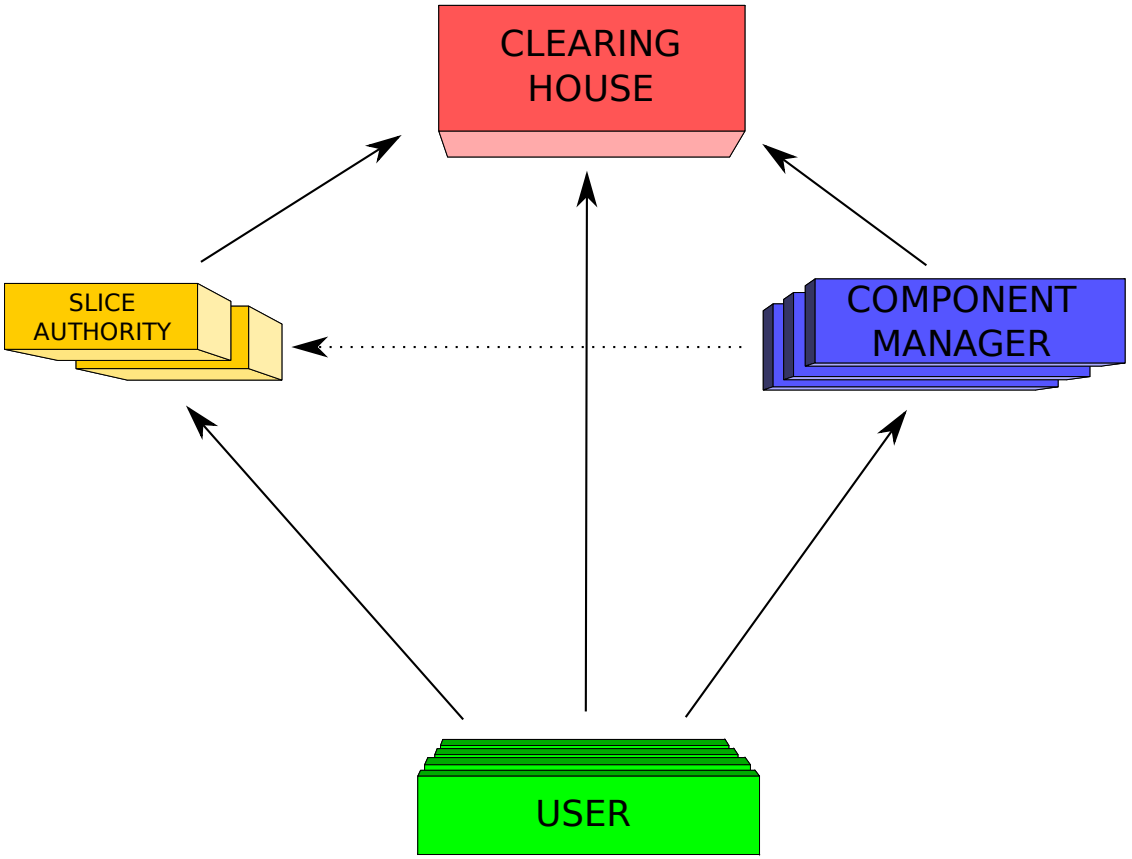
Gary Wong
University of Utah

17 November, 2009

Introduction

- The ProtoGENI XMLRPC interface
- Certificates and credentials
  - Accessing a component manager
- Components and rspecs
  - Discovering and specifying resources
- Slices and slivers
  - Allocating and operating components

Federation overview



XMLRPC Interface

- XMLRPC over HTTP over TLS (SSL)
  - Both client and server must identify and authenticate
  - Self-signed root certificate for each site
- URLs available from the clearing house
- Request: standard `methodName` and `params` elements
- Response: `methodResponse` and `params` elements; `params` includes:
  - `code` (0 for success; various non-zero error codes defined)
  - `value` (result of operation; type varies)
  - `output` (human readable description)

Querying the CM version

```
struct result = GetVersion();
```

```
int result.api;
int result.level;
string result.input_rspec[];
string result.output_rspec;
```

- minimal component managers implementing the current version will return **api** 2, **level** 0, **input_rspec** (0.1) and **output_rspec** 0.1.

Object names

- All ProtoGENI objects are named with a URN (see GMOC Proposal)
- Of the form:

  `urn:publicid:IDN+authority+type+identifier`

- Examples:
  - `urn:publicid:IDN+planet-lab.org+user+cviecco`
  - `urn:publicid:IDN+planet-lab.org+node+pl2.ucs.indiana.edu`
  - `urn:publicid:IDN+emulab.net+slice+mytestslice`
  - `urn:publicid:IDN+emulab.net+authority+sa`

Credentials

- Simple XML documents (with XML-Dsig)
  - Certificates
  - Privilege descriptions
  - Signatures
- Can be issued by any authority (or even user)
  - Typically a slice authority
- Secure proof of a principal's permissions
  - Distinct from proof of identity (TLS)
- Utah will provide library support
- Policy is deferred to component managers and site admins

rspecs

- Three* varieties:
  - Advertisements
  - Requests
  - Manifests
  - (and Tickets)
- Body is an annotated list of components
  - Nodes
  - Interfaces
  - Links

*okay, four

Listing resources

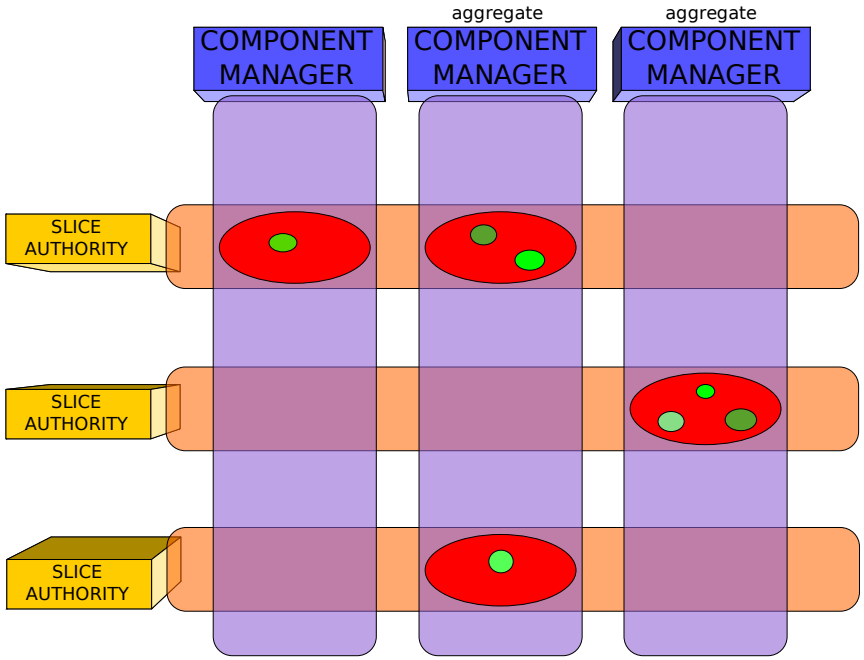
```
result = DiscoverResources( string credentials[],
 boolean available, optional boolean compress );
```

- **result** is an advertisement rspec
- **available** requests only components not currently in use
- **compress** is a hint that the client would like **deflate** compression applied

Slices

- Containers for resources
- Potentially federation-wide
- Slice authorities issue slice credentials
  - Might subsequently be delegated

Slivers



- Local element of a slice
- Tightly bound to one slice and one component manager
- We unify component slivers and aggregate slivers as much as possible

Object information

```
struct result = Resolve( string urn,
                         string credentials[] );
```

- **result** structure will vary depending on CM and object type

- example:

```
<struct>
  <member><name>urn</name><value>...</value></member>
  <member><name>hostname</name><value>...</value></member>
  <member><name>rspec</name><value>...</value></member>
</struct>
```

Allocating resources

```
struct result = CreateSliver( string slice_urn,
    string rspec, optional struct keys[],
    string credentials[] );
```

```
string result.sliver;
string result.manifest;
```

- `result.sliver` is a sliver credential
  - includes the sliver name (URN)
- `result.manifest` is an rspec with details of resources

Retrieving a sliver credential

```
string credential = GetSliver( string slice_urn,
                               string credentials[] );
```

- typical use is to give a slice credential and obtain a sliver credential

Sliver health

```
struct result = SliverStatus( string slice_urn, string credentials[] );
```

```
string result.state;
string result.status;
struct result.details[];
```

- `state` is the *administrative* condition (`started` or `stopped`)
- `status` is the *operational* condition (`ready`, `notready`, `changing`, `failed` or `unknown`).
- `details` includes the individual conditions of any child components (useful for aggregates).

Cleaning up

```
int DeleteSlice( string slice_urn, string credentials[] );
```

- deallocates the sliver (on a single CM)

- it's **DeleteSlice** and not **DeleteSliver** for obscure reasons

Rebooting

```
int RestartSliver( string slice_urn,
                   string credentials[] );
```

- unspecified whether state is preserved or lost

Requesting more time

```
int RenewSliver( string slice_urn, string valid_until,
                 string credentials[] );
```

- initial expiry time was given in the rspecs during sliver creation
- no other facilities for modifying reservations on minimal CMs

Emergency stop

```
void Shutdown( string slice_urn, boolean clear,
               string credentials[] );
```

- either shuts down a slice, or **clear**s the shut down state
- no operations are permitted on a shut down slice except **Shutdown** itself