Dimma

Dynamically Loaded Protocols

Simon Waterhouse

sww@ansa.co.uk

Objective

- Minimise Dimma footprint:
  - each protocol in its own shared object (DLL);
  - only load protocol if it is required;
- Facilitate addition of new protocols:
  - dynamically...
  - … with no DIMMA kernel rebuild …
  - … and no application rebuild required;

Protocol Independent Stubs

- Dimma v1.0 stubs were not truly protocol independent:
  - assumptions about the RPC protocol representation of operation and termination names;
- Dimma v1.2 introduced new classes to support mapping from name to number:
  - odp_nameTable;
  - odp_NameTableEntry;
- …stubs are now protocol independent;

Protocol Loader

- A Capsule has a single Protocol Loader;
- Protocol Loader can load:
  - pre-configured protocols (currently IIOP, AnsaFlow);
  - ‘plug and play’ protocols added at run-time;
- In Dimma 2.0, add a new protocol by:
  - describing it in a text file;
  - building a shared object containing the protocol implementation;
  - restarting the Dimma application;

To specify a new protocol

- Create a file containing:
  - Protocol name - from which the name of the shared object will be derived (platform specific);
  - Protocol type (flow or rpc);
  - Protocol number - the IOP tag for the protocol (cf. TAG enumeration in file ‘IOP.hh’);

```
#
# This is an example protocol description file
#
myRPCProtocol          RPC     2
myFlowProtocol         Flow    1001
```

To specify a new protocol(2)

- Define an environment variable:
  - DIMMAPROTOCOLS - pathname for file containing the protocol description;
- Ensure the new protocol shared object is accessible to the application (i.e. use LD_LIBRARY_PATH on Solaris);
- Restart Dimma application;

Which Protocol is loaded?

- Explicit binders get to choose (see EngineeringQoS);
- Implicit (server) binder:
  - create a binding for each known (RPC or Flow) protocol;
  - export an IOR containing a tagged profile for each protocol;
- Implicit (client) binder creates a binding for the first recognised tagged profile in the IOR;
- Or write your own implicit binders!