Middleware Labs: CORBA

Petr Tůma Vojtěch Horký Antonín Steinhauser
Vladimír Matěna

March 20th, 2019

Department of Distributed and Dependable Systems

General Information

- Task description is on the web
- IDL specification, mapping to C++
  - http://www.omg.org/spec/CPP/1.3/
  - http://www.omg.org/spec/CPP11/1.1/

CORBA implementations

- omniORB
  - Download from the website of the course
  - Upack into `~/omniORB`
  - **Or** use your distribution-provided omniORB
  - Update Makefile as necessary...
  - **Or** compile from source
  - `https://sourceforge.net/projects/omniorb/files/omniORB/omniORB-4.2.2/`
- TAOX11 (mapping to C++11)
  - Uses C++11-specific features
  - Download from `https://swsupport.remedy.nl/`
  - There are plenty of installers, at least Fedora 17 seems to work

Example (C++, omniORB)

- `Makefile` for client and server
- `simple.idl`
  - `simpleSK.cpp`, `simple.h`
- `server.cpp`, `simpleSK.cpp`, `simple.h` → `server`
- `client.cpp`, `simpleSK.cpp`, `simple.h` → `client`

Example (C++11, TAOX11)

- build.sh runs MPC and make on the generated Makefiles
  - Need for initialization, then call just make
  - Extra clean.sh for removal of all generated files
- setenv.sh used to setup environment for taox11 execution
- source ./setenv.sh # use this once before running server and client
- simple.idl
  - simpleC.cpp, simpleS.cpp
  - simpleC.h, simpleCP.h, simpleS.h, simpleSP.h
- server.cpp, simpleC.cpp, simpleS.cpp, ... → server
- client.cpp, simpleC.cpp, ... → client

The Task

Implement a client talking with server we provide. . . [1]

$\rightarrow$ Hello, I am . . .
$\leftarrow$ Hi, your key is . . .
$\rightarrow$ Hello, I am . . . and my key is . . .
$\leftarrow$ Hi, wait until I am ready, please . . .
$\rightarrow$ Tell me your status . . .
$\leftarrow$ Here it is . . .
$\rightarrow$ Part of your status is . . .
$\rightarrow$ Bye.

[1] . . . and also reimplement the server by yourself.

Client Implementation

- IDL is in `master.idl`
- Reuse files from the example
  - In `Makefile` just update the IDL file
- Client code goes directly into `main()` in `client.cpp`
- Pass the input parameters (IOR, key) as command-line arguments
- Report issues, ask questions when unclear
- Read the slides
  - To avoid or solve typical problems ;-)
  - Return to them after you have read the instructions

Server Implementation

- Preferably in C++
  - Talk to us if you wish to implement it in different language
  - Not in Java (half of the tasks is in Java already)
- Mimic the behaviour of our server reasonably
  - Use common sense

Submission

- In C++ and omniORB or C++11 and TAOX11
  - Server might be in different language (talk to us first)
- By e-mail (deadline is on the web)
- The submission shall be easy to start
- Do not send any generated files (but send the build script)
- Brief README never hurts
  - Especially if your server behaves slightly differently

Notes

- Print what the client does to standard output
  - `cout << "Connected, peer " << peer`
  - `      << ", key " << key << endl;`
- Use `sleep(1)` when waiting for idle
  - Do not overload the server
- Be careful with memory allocations
  - CORBA may deallocate (in)out parameter
  - E.g. inout strings pass as copies
    - `CORBA::string_dup("foo")`
    - Or use helper class ("smart pointer")
- Server code can be executed in parallel

Common Problems

- `MARSHAL` exception when calling `connect()` (or `ping()`) for the first time
  - Is IOR string really correct?
- Class (e.g. `String_out`) cannot be instantiated because it has private constructor
  - Deriving correct mapping of IDL solely from function signatures in `master.h` might be misleading!
  - Parameter types are for omniORB implementation or for server, not for client
    - Correct client types are automatically type-casted

Department of Distributed and Dependable Systems D3S

Notes for TAOX11

Evaluation license

- Must be placed to `taox11` root (unpacked)
- Limits the execution time of the application
- Must be on the path of the application running
  - In the working directory
  - In the `RI_LL_LICENSE` environment variable

Building and compiling

- Use the MPC workspace creator `mwc.pl`, don't try to write the `Makefile` manually
- **Or** use the BRIX11 toolset