Petrus Repo
petrus.repo@cs.helsinki.fi
4.5.2010

PSIRP CODE CAMP

REPORT OF TASK A1

INTRODUCTION

Implementation of my work is object oriented. I would really like that Python API of libpsirp be improved in an object oriented direction. For the moment the API is strongly influenced with functional programming style.

I also had some difficulties in the beginning. Biggest was a technical problem with libpsirp API, which took most of my time in the beginning. The cause could a bug in the API. It is described below.

TEST DRIVE

The program can be tested as follows:
1) Start 1..N Subscribers with "main_subscriber.py" before Publisher.
2) Start a Publisher with "main_publisher.py".
3) Start 1..N Subscribers after publisher.

THOUGHTS ON IMPROVEMENTS

PSIRP's API methods should be queried from a specific namespace or from an object. Now functions such as create() and sub_s() are in global namespace. I would prefere e.g. calling a constructor Publication() instead of an "independent" function create(), or using any instance method of an object.

Why the Python programmer has to take care of a publishment's length?
Why can't PSIRP adjust buffers dynamically?
Why not to provide an object which could calculate the length internally?
Taking care of any buffer length in a high level language feels a little bit inappropriate because then you need to concentrate on "how to do" instead of "what to do" (the business logic).

Could there be a "replace"-function built in the API which would replace old published content with new content without having to deal with buffer lengths manually?

Why not to provide an event-listener as a built-in method in API?
Now I just copy-pasted event_example.py to its own module because I did not want to reinvent the wheel. API could provide the code and yield a callback to allow custom behaviour for e.g. handle_event().

If Sid and Rid are strings, the following method of PubSubKQueue fails silently and does not work. It should raise an exception and tell that parameters were incorrect.
Parameters can only be atoid(sid) and atoid(rid).
- FAILS: register_advance_subscription("::aa", "::bb", [..])
- WORKS: register_advance_subscription(atoid("::aa"), atoid("::bb"), [..])