marcopolo Documentation

Publicación 1

Diego Martín

07 de April de 2015

Índice general

Contents:

CAPÍTULO 1

Fundamentals

MarcoPolo (named based on the swimming pool game [1]) is a simple utility designed to allow hosts to publicly share the available services inside a local-area network, allowing multiple grids of nodes to work in the same net without any net interference and hosts to work on different grids simultaneously.

MarcoPolo is built upon UDP-based low-level multicast sockets. This reduces the consumed bandwith due to the usage of multicast instead of broadcast messaging and the lack of a TCP three-way handshake except when proven necessary (as seen later).

The architecture of the system is based on two independent modules, inspired on the dynamics of the Marco Polo game:

- A **Marco** instance, bound to each application in need of access to the protocol. It is in charge of creating commands to multicast in the network, and works very much like a DNS resolver (through a local socket on 127.0.1.1).
- A **Polo** instance running as a daemon process listen continually for incoming multicast in a certain (configurable) group(s), sending responses to a request when this satisfies a set of conditions (same multicast group, request for a service which is offered by the node...).
- All messages are (though only temporarily) codified as JSON messages and sent encoded as UTF-8 strings.
- A set of configuration files to customize the functionality of the daemons.
- A set of bindings for Python, C/C++ and Java.

[1] http://en.wikipedia.org/wiki/Marco_Polo_%28game%29

CAPÍTULO 2

Configuration

The standard configuration allows MarcoPolo to run pretty much in any small network. However, there are a certain conditions which will require a little configuration by the end-user:

- Different grids on the same network: MarcoPolo is bound to a certain default multicast group (224.0.0.112). In the event that this group is already on use on the system by other MarcoPolo grid or any other application some problems may occur. Particularly on the first case, since MarcoPolo simply does not reply to any bad-formatted message (and the odds of a JSON-based multicast application running on the same address are quite small). On the file /etc/marcopolo/marcopolo.conf the parameter MULTICAST_ADDR can be set to any compatible IPv4 multicast address. (See [1] for a reference).
- Multiple local-area networks: Multicast packets can be routed up to a global level (if the selected group is allowed for such task). The parameter HOPS sets the TTL (Time To Live) of the package to distribute the package beyond the local designated router. **Important**: use only with a supported MULTICAST_ADDR value.

2.1 Services

The offered services may be configured in two ways:

- A file in the services/ directory with the following skeleton:

```
{
 "id":"<unique identifier>",
 "version":"<version>"
 "startup-command": "<command>"
 "shared-only": "<list of multicast groups separated by commas>"
 "allows-copy": "<yes/no>" //Allows copy of the .mar file
 "requires-auth": "<yes/no>" // Some services will only be shown to authenticated requests
}
```

The script 'poloservice' offers an interactive method to create a service file and publish it.

- By using the *register()* and *deregister()* functions during execution time.

[1] IANA Guidelines for IPv4 Multicast Address Assignments http://tools.ietf.org/html/rfc5771#page-4

CAPÍTULO 3

Indices and tables

- genindex
- modindex
- search