OdysseusNet Projects

There are many project, that form together OdysseusNet. Here we will give a short overview. We omit the de.uniol.inf.is.odysseus part of the project and will only show non feature projects.

Remark: The are two different ways to use OdysseusNet.

- All nodes are OdysseusNet nodes
- A least the master node is an OdysseusNet node and the other node are default Odysseus nodes (Remark: Even if the worker are OdysseusNet nodes, they are treatet like default Odysseus nodes, i.e. no extra worker capabilities are used).

net

In this project all the base and common classes can be found.

net.communication

Here all basic interfaces for the inter node communication can be found. Odysseus currenlty provides two ways to communicate. Message based (here all nodes must be OdysseusNet nodes and all node must be part of the same subnet) and REST-based (only the master node must be an OdysseusNet node, the other nodes are default Odysseus nodes). Only one communication kind can be used at the same time and is configured with the `net.node.communicator` configuration. The main interface here is IOdysseusNodeCommunicator.

net.communication.object

This project implements the message based OdysseusNet communication. Here all nodes must belong to the same subnet, as the worker will send messages to the master. More concrete: Internally, OdysseusNet internally communicates with message object. This layer just sends the message objects to other Odysseus node and receives messages from other nodes. net.connect (see below) is used for this.

net.communication.rest

This project allows a master to communicate with default Odysseus nodes by using the REST interface V2 of Odysseus. The internal messages of OdysseusNet are caught and translated to REST class and the results are send as messages again. The configure this rest based communication, use `net.node.communicator='rest'` in the OdsseusNet configuration.

net.config

This project handles the access to the net config file.

net.connect

This project is used for the object based communication via tcp sockets. It has some associated project:

- net.connect.select.all: Group ids of nodes are ignored.
- net.connect.select.group: Handle different groups of nodes.
- net.connect.socket: Basis socket communication between OdysseusNet nodes.

net.console

This project provides OSGi console commands for OdysseusNet

net.data

All distributed source handling is done here. This is not tested with the REST based communication yet.

net.data.console

A console for distributed source handling.

net.discovery

The discovery of nodes is handled here. There are currently three discoverer implemented:

- net.discovery.broadcast: Only for object based communication. Uses a broadcast to look for existing nodes. Nodes will reply to a broadcast message. (If the workers are OdysseusNet, this can be combined with the REST communication.)

- net.discovery.multicast: Only for object based communication. Uses a multicast to look for existing nodes. Nodes will reply to a multicast message. (If the workers are OdysseusNet, this can be combined with the REST communication.)
- net.discovery.iplist: The discoverer reads nodes from a config file. The nodes can be any kind of nodes.

net.ping

This project handles node alive checking with ping messages.

net.querydistribute

This project handles the basics of query distribution (see Distributing Queries). A large set of associated projects implement different strategies:

- allocate.*: The different strategies for query allocation. Other strategies should be implemented similar.
    - direct
    - querycount
    - roundrobin
    - user
- modify: The modify strategies.
    - replication
    - TODO: partition
- partition: The strategies that handle the way to partition the query plan can be found here:
    - operatorcloud
    - query (cloud)
- postprocess: Different postprocessing strategies
    - calculations
    - discardreplicates
    - localsink
    - localsource
    - merge
- proprocess: Preprocessing strategies
    - sources

net.rcp

Any frontend related stuff can be found here (e.g. node view)

net.recovery

TODO

net.resource

Providing information about free hardware resources.

net.sources

The basics for distributed data source management

net.startup

Provides some startup handling

net.update

Handling of node updates. This works only for object based communication. When using docker, this should not be used.

net.util

This package provides some helper classes..