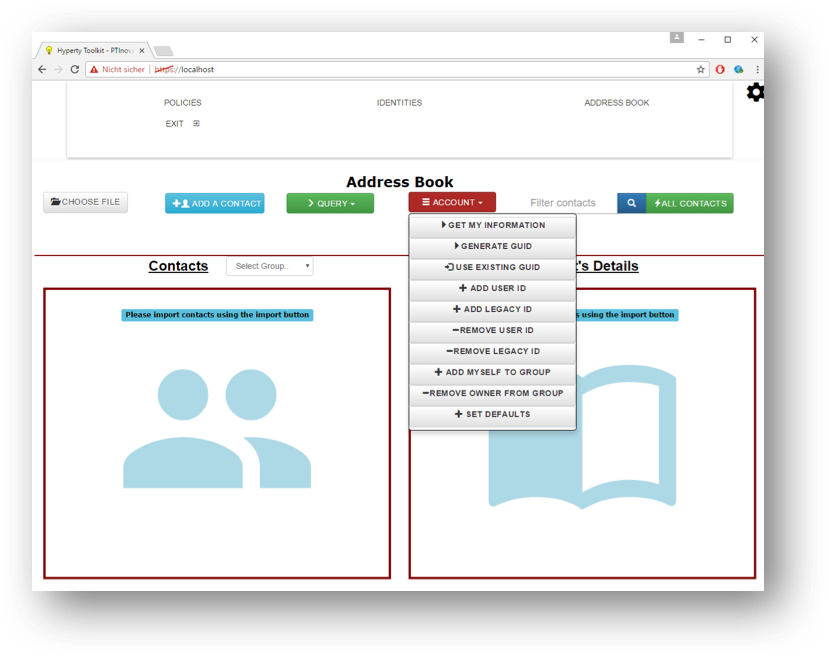
## Cross-Domain Discovery evaluation

### Short description

This section describes the usage of components involved in the IMaDS discovery and registry framework [1][2]. Using the IMaDS framework, users can connect to other users' devices using a immutable GUID. The framework uses three components, being Discovery, Global Registry, and Domain Registry, while the ReThink runtime provides a GUI component to store and manage contacts in an addressbook application

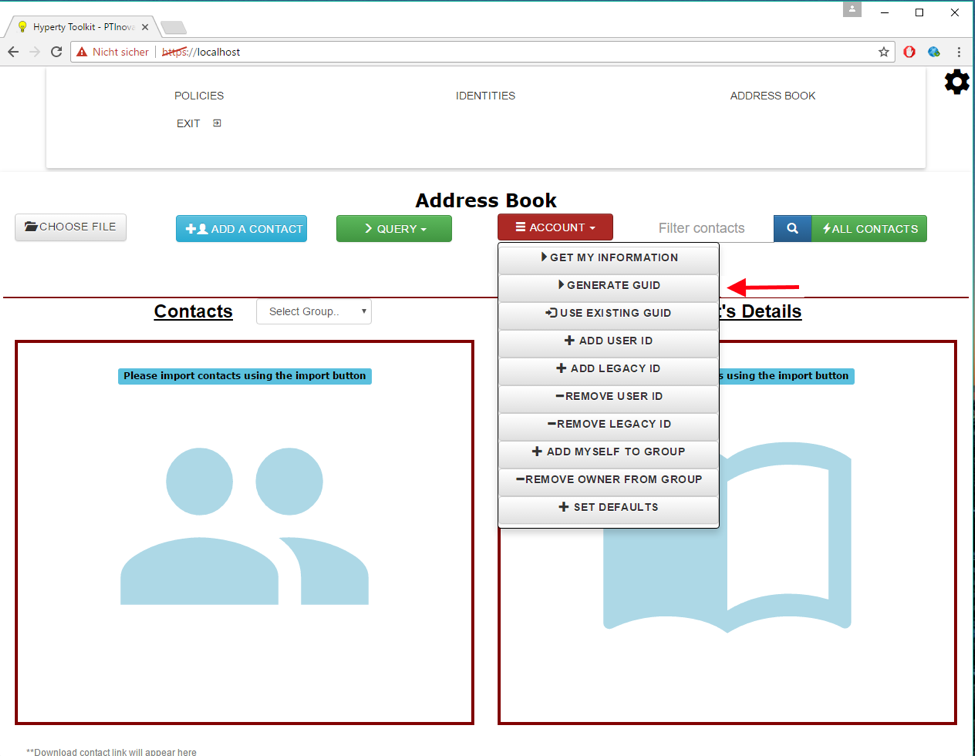
### Functional tests procedures

The Addressbook is a web based application included in the ReThink runtime that allows users to create and maintain one's identity. The Addressbook is reachable in the browser at localhost as part of the admin interface of the ReThink runtime. Its main screen shows a list of contacts and means to query for and add users:

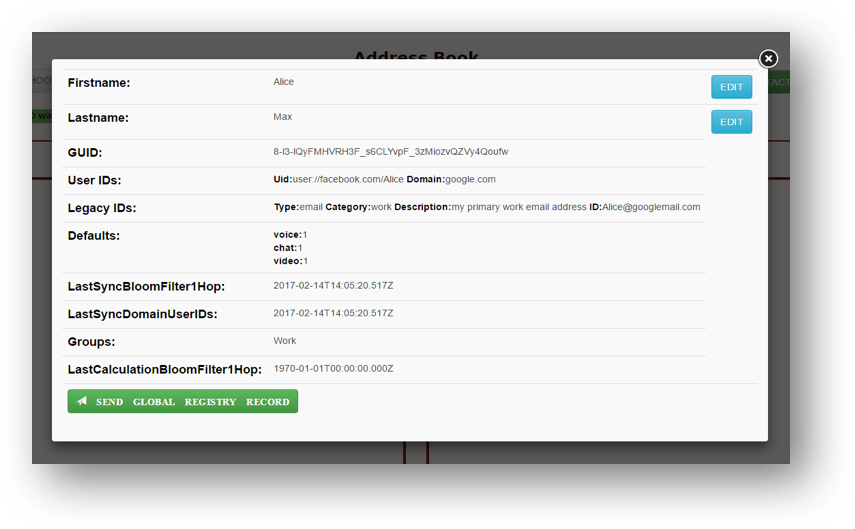


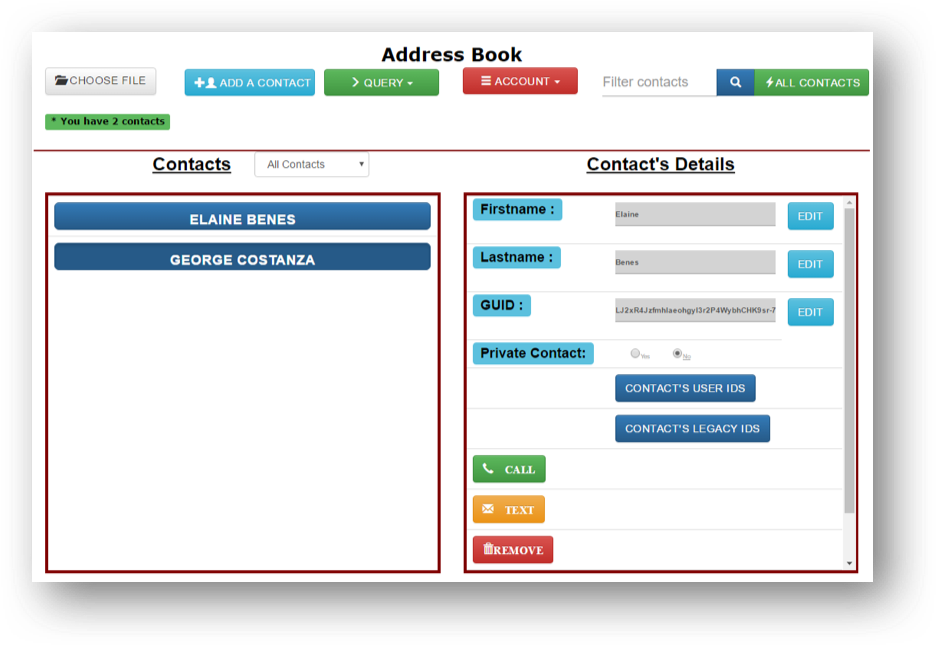
Addressbook Create GUID

At first use, a new GUID needs to be created and registered in the Global Registry service. This is automated via the option "generate GUID" in the "Account"-menu. The process registers a new GUID and publishes it in in the Global Registry service along with information specified in the registration process.

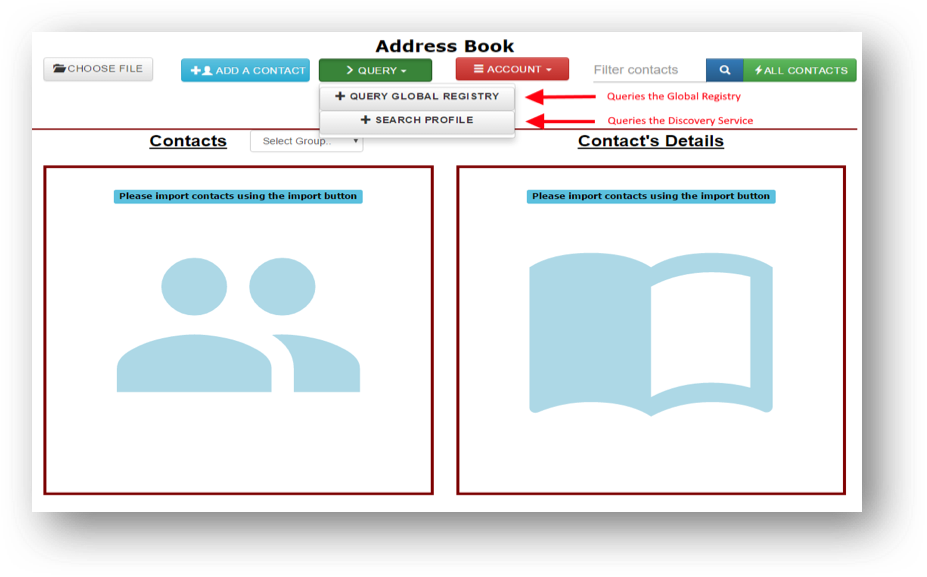


Addressbook Info

To view and edit one's contact details, users can use the "get my information" function that displays a dialog with all details of the user's account. Changes can be made directly in the dialog and the changed account details can be directly sent to the Global Registry: 

When selecting contacts in the Addressbook, available information for the selected contact is shown. This includes the username and ids: 

The Addressbook further allows to search for users either by a known GUID or arbitrary contact details:



Addressbook Query

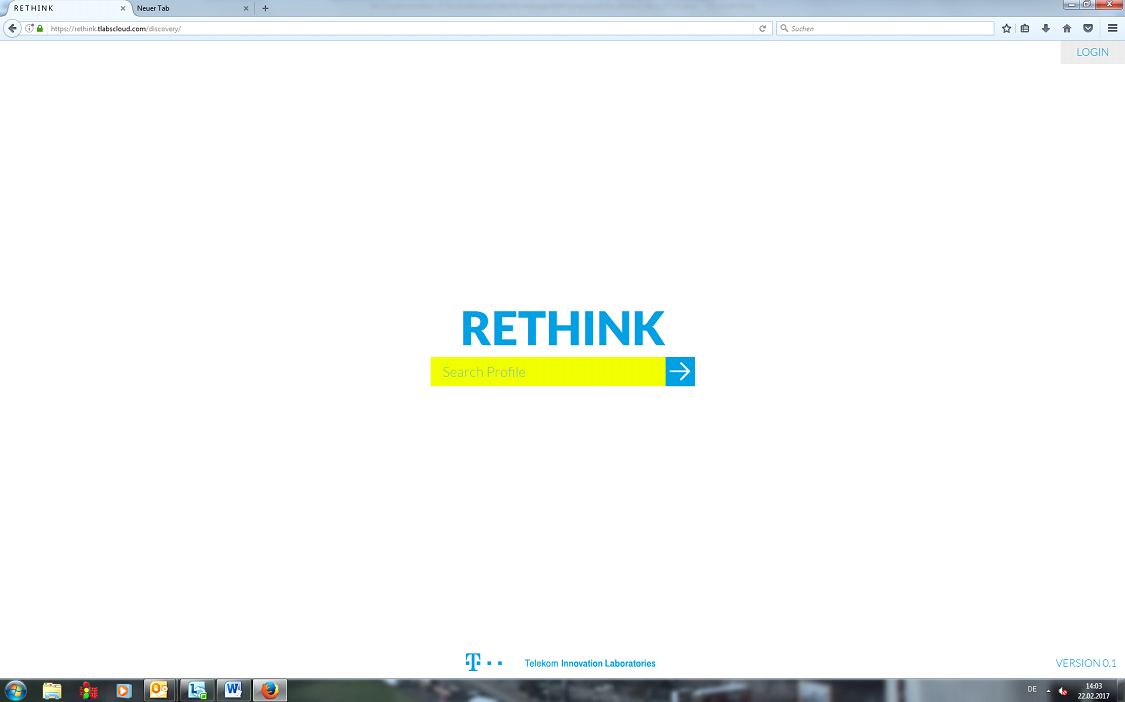
The DISCOVERY SERVICE further provides a standalone web interface to enable users and other services to search for users. People are used to use search engines like google when they want to find information about a certain topic. In the same way, they can search for persons or also devices in reTHINK. They formulate a search query like “Michael Mueller T-Labs” or “reTHINK Project” in order to find one or more so called profiles of a person or a device. Although this service is easy to use like a classic search engine, there are fundamental differences.

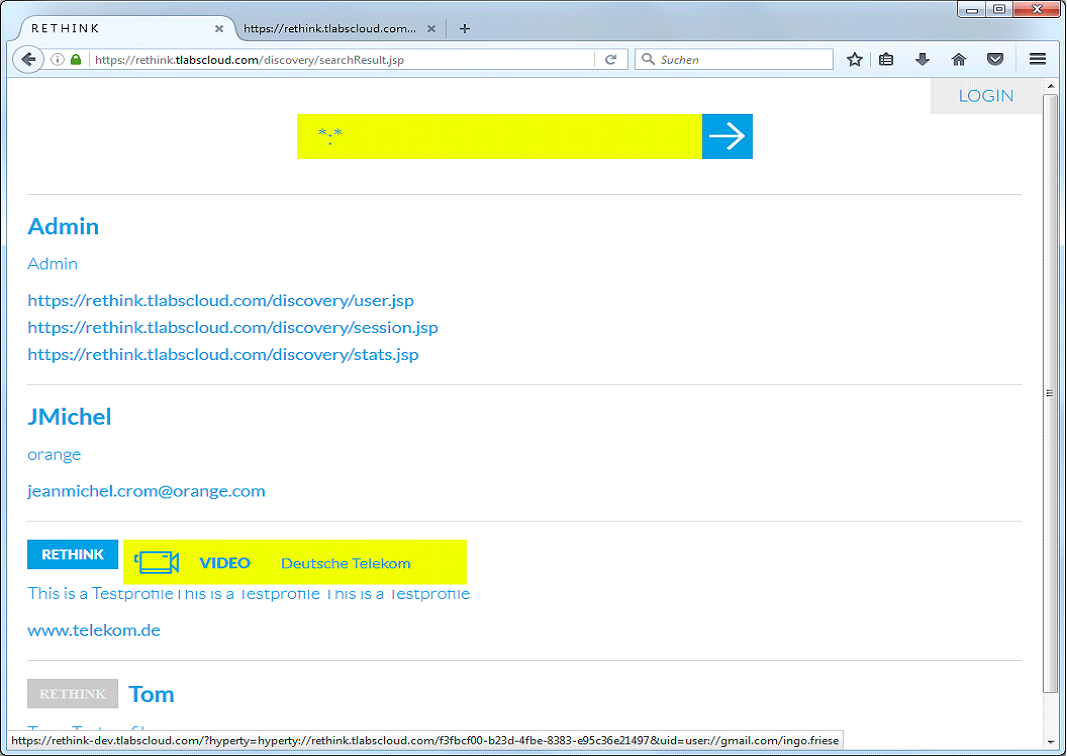
Source of query results: Search results are profiles that are written by the user that wants to be found. Results are not crawled or scanned in the Internet. Every user that wants to be found or that wants his devices to be found, can create one or many profiles within the discovery service. These profiles stay fully under user control.

Policy-based visibility: Every profile can be configured with a certain visibility. So the owner of a profile can configure which user or which group of users can see his profile.

Interfaces

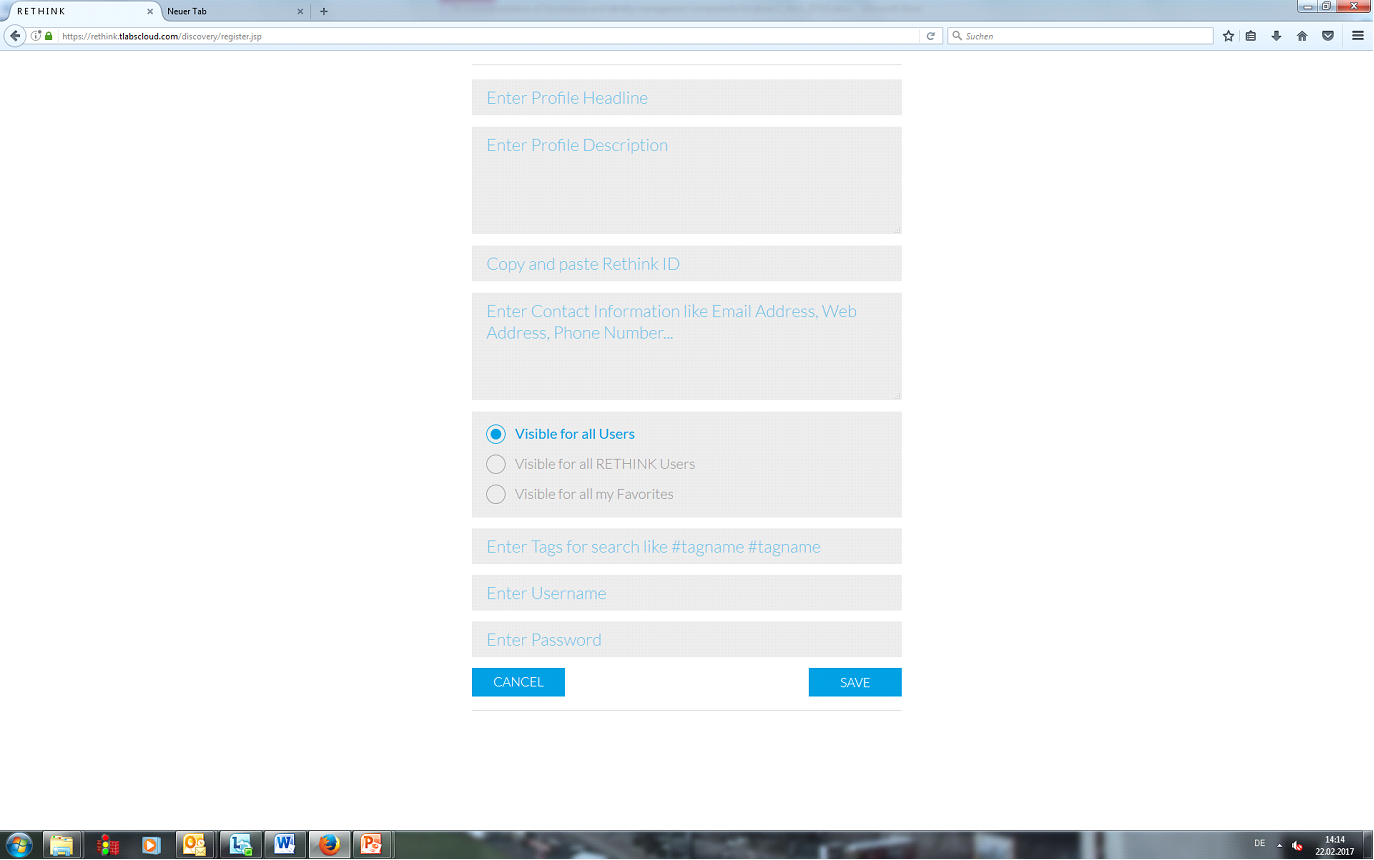
Web-GUI: The reTHINK discovery has a Web-Interface for users that want to use reTHINK discovery with their browser, like they are used to from a classic search-engine. Furthermore the web-GUI enables to create and manage profile accounts.

 ' 1) Plain web search: The user can go to the reTHINK Discovery Website and search for users or devices. The search results are so called profiles. They have a headline and some text for description. They might have hashtags describing certain topics, communities, locations etc.. Profiles can also contain communication endpoints like e-mail address, phone number, websites, facebook- or linkedin profile URLs.



Discovery GUI search results

1. Create his own account: Every user has the possibility to create his own account with the reTHINK discovery service.



Discovery GUI profile registration mask

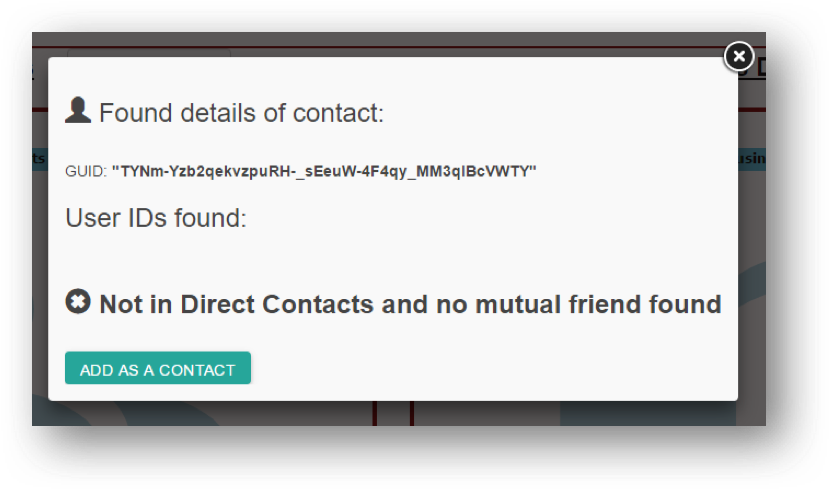
REST-API

The discovery service offers also a REST-API for search queries, that can be used from other services or applications. The interface can be "pinged" by e.g GET https://rethink.tlabscloud.com/discovery/rest/discover To search eg. for "Hans Telekom" call GET https://rethink.tlabscloud.com/discovery/rest/discover/lookup?searchquery=Hans+Telekom The answer is a JSON Object like for Example:

{  
"instanceID":"telekom1",  
"responseCode":201,  
"searchString":"Hans+Schmitt",  
"results": [  
{  
"resultNo":0,  
"instanceID":"telekom1",  
"hashtags":"T-Labs Telekom",  
"description":"Hans Testprofile text text text",  
"rethinkID":"REAyDT-tYQI2u1km9LgYj05zb1hLmy\_\_XlIN5B1LWUQ",  
"headline":"Hans",  
"contacts":"Hans.Schmitt@telekom.de",  
"hasrethinkID":"true",  
"hyperties": [  
{  
"url":"hyperty=hyperty%3A%2F%2Frethink-dev.tlabscloud.com%2Fd2aad579-a213-4400-9e24-af2b23dbaed8",  
"userID":"uid=user:\\gmail.com\hans.schmitt",  
 "media":"VIDEO",  
 "provider":"Deutsche Telekom"  
},  
{  
"url":"hyperty=hyperty%3A%2F%2Frethink-dev.tlabscloud.com%2Fb71cd853-f657-4b6a-919b-07916bd3d50e",  
 "userID":"uid=user:\\gmail.com\hans.schmitt",  
 "media":"VIDEO",  
"provider":"Deutsche Telekom"  
}  
]  
 } ] }

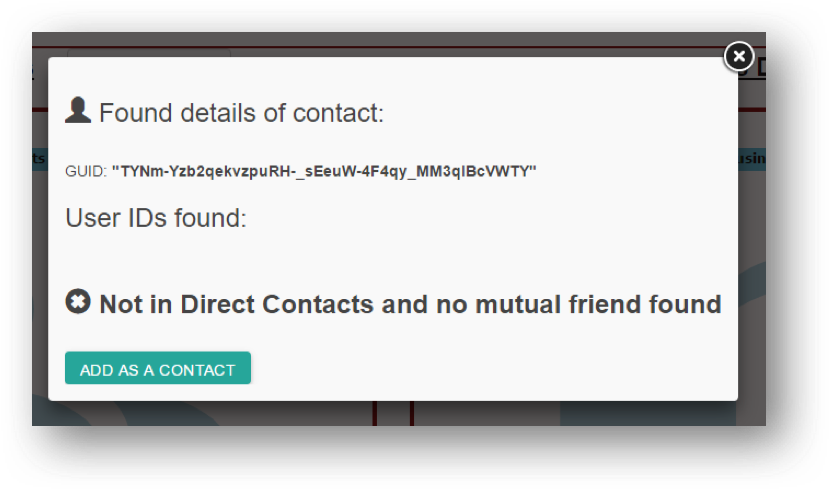
The result contains, beside the content, also the instanceID. This instanceID is used to distinguish between different Discovery service instances. Furthermore, the endpoint of all hyperties that are currently running are shown, including supported media type and the regarded service provider as well as the used identity aka. userID.

In case the GUID is not known, the Discovery service of the ReThink framework can be queried. Here, all found contacts matching the search query are returned and can be added directly to the contact list:



Discovery Result

In case the GUID of the contact is known, the Addressbook application directly contacts the Global Registry service. Information retrieved from the Registry service is displayed in a dialog. The contact can then be directly added to the contact list:



Addressbook Result

The Global Registy provids an association between each user identity (GUID) and the service provides with which he is registered. This is a relationship that changes infrequently. However, the same does not hold true for the information regarding the devices being used by the user to access each service provider's services. As devices change networks and are turned on and off, reachability information must be updated frequently. As such, each Service Provider run its own instance of a Domain Registry, where a mapping between each user and the services (hyperties) being used in each of the user's devices is kept. The runtime of each device is responsible for keeping this information up to date. The interaction with the Domain Registries is transparently performed by the Runtime on each device. The user is not aware and does not have to performe any action.

While the example described above covers manual discovery and handling of user identifiers, resolving is usually carried out automatically. Here, a user specifies a GUID to contact, e.g. selected from the Addressbook application, which he wants to connect to, e.g. call. The IMaDS framework then automatically resolves a GUID to the user's UserIDs and further to this user's active Hyperty instances (using the suitable Domain Registry). Following this process, a connection is automatically initialized between two users.

A typical usage scenario of the IMaDS services comprise a user Alice, who wants to connect to another user Bob, who is registered in one or more service domains. If required, Alice would query the Discovery component to search for Bob using facts that describe Bob. By resolving Bob’s GUID via the Global Registry, she receives a list of service domains Bob is registered with. As each UserID in the returned dataset also comprises information about the respective service provider’s Domain Registry, Alice can now query the Domain Registry for Bob’s communication endpoint and establish a connection.

### Non-functional evaluation

The runtime communicates with the Discovery Service, Global Registry and Domain Registry using the Discovery Lib, a runtime library that provides a single API for accessing the 3 services. The Discovery library allows the discovery of remote Hyperties or Data Objects by using different criteria. To evaluate this component, the most complex query available was used, which discovers Hyperties or Data Objects by profile data. This query translates the profile data into GUIDs by querying the Discovery Service. Then, for each found GUID, it executs a query to the Global Registry to get all the user IDs and domains associated. Finally, for each user ID and (Service Provider) domain pair, a query to the corresponding Domain Registry is executed, to get the registered Hyperties or Data Objects.

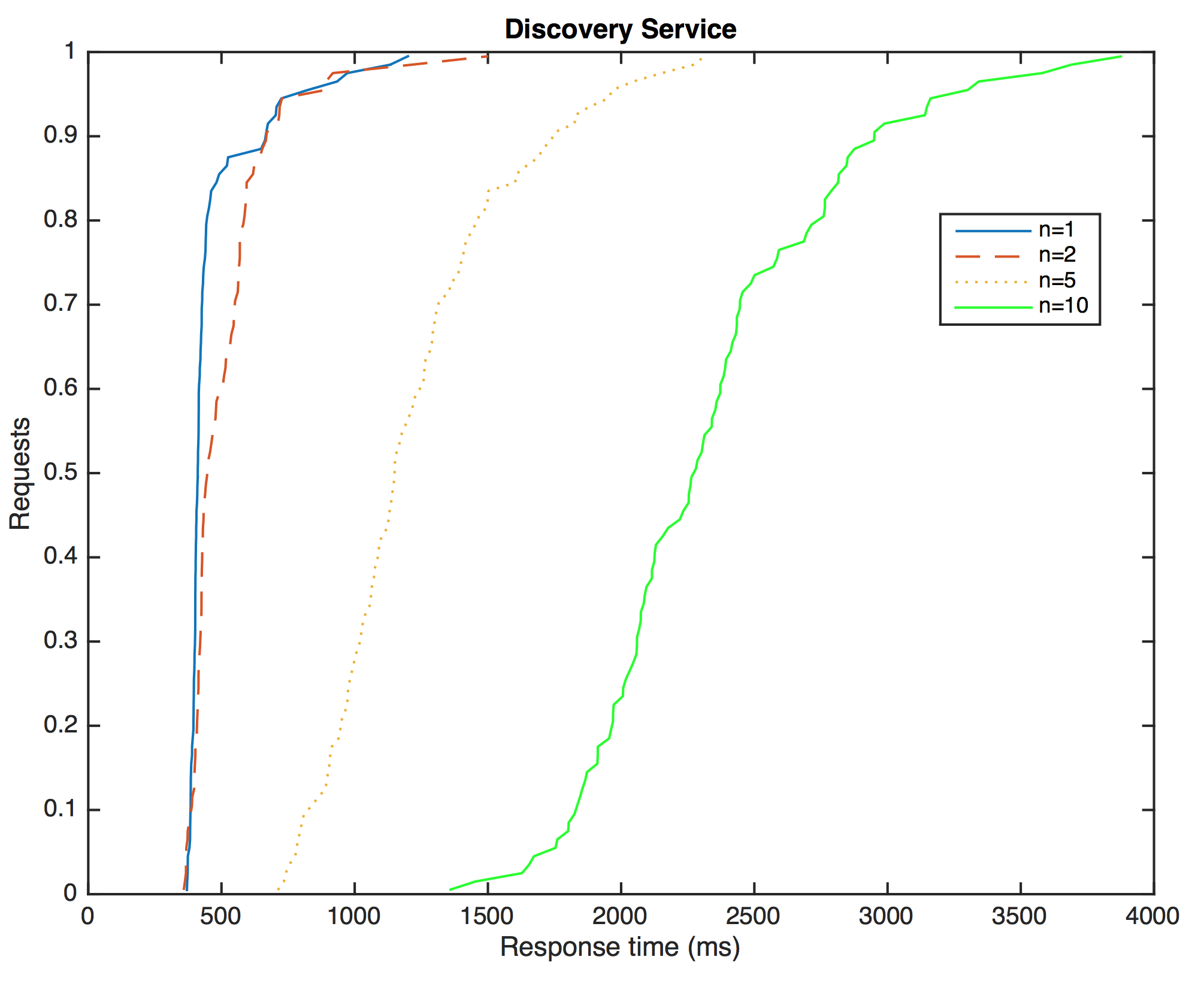
The goal of these tests is evaluate the performance of the combined IMaDS from the point of view of the user, by querying all three, in succession, from the runtime.

The methodology used to evaluate the Discovery library was to develop a web application that loads an Hyperty programmed to use the discovery service using profile data. For the web application to work properly and discover the Hyperties associated to the profile data, we had to create an account in the Discovery Service with a GUID registered in the Global Registry. A Service Provider domain and an user ID were associated to that GUID and an hyperty registered in the associated Domain Registry. Then, we run the tests, collecting four metrics in each execution:

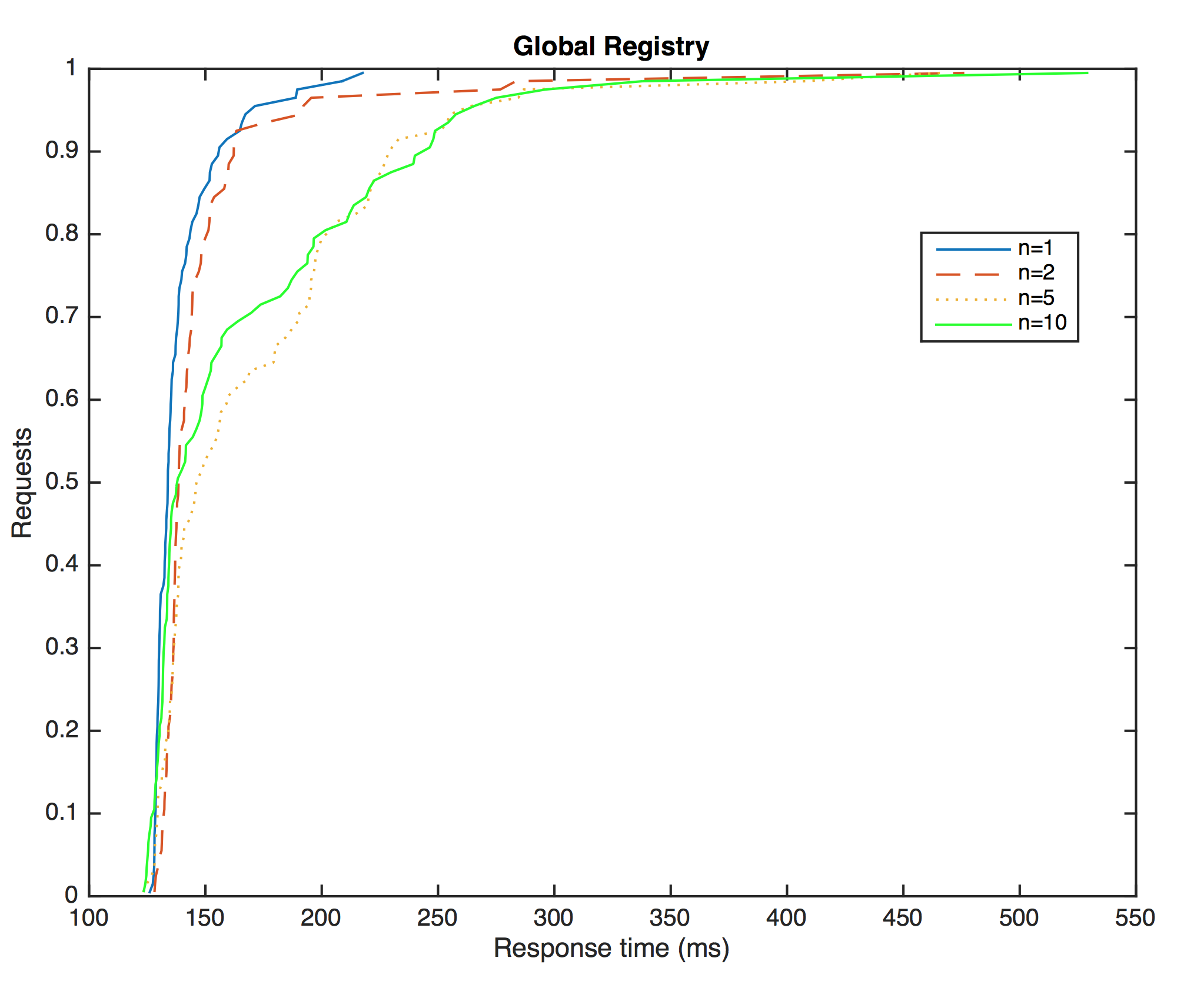
* Time necessary to complete a query to the Discovery Service;
* Time necessary to complete a query to the Global Registry;
* Time necessary to complete a query to the Domain Registry;
* Total time spend in the discovery by profile data: sum of the previous three steps.

The tests were repeated one hundred times. The client was run on a MacBook Pro, using the Chrome Browser, connected to INESC-ID's network. The Domain Registry was a part of Altice Labs test bed. The Discovery Service is the one provided by T-Labs. The Global Registry is the one shared by all testbeds and it was interfaced via Altice Labs testbed.

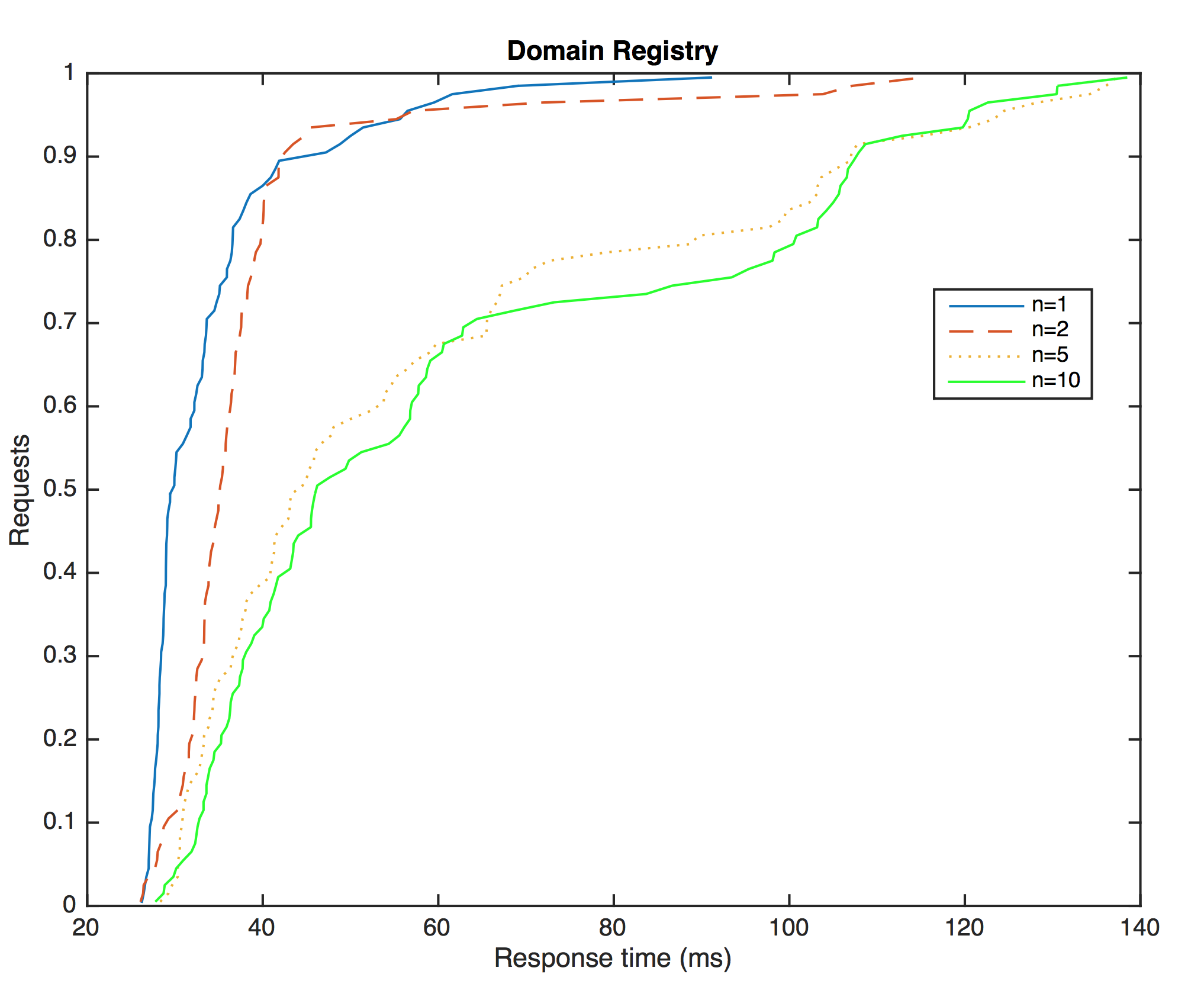
In order to test the performance, we varied the number of concurrent requests, using 1, 2, 5, and 10. As this is run from a single runtime, concurrent requests are not expected to be frequent and 10 concurrent requests is a very unlikely event. Below is a chart for each metric. Results are presents as a cumulative distribution function (CDF), with a line for each number of concurrent requests (n).



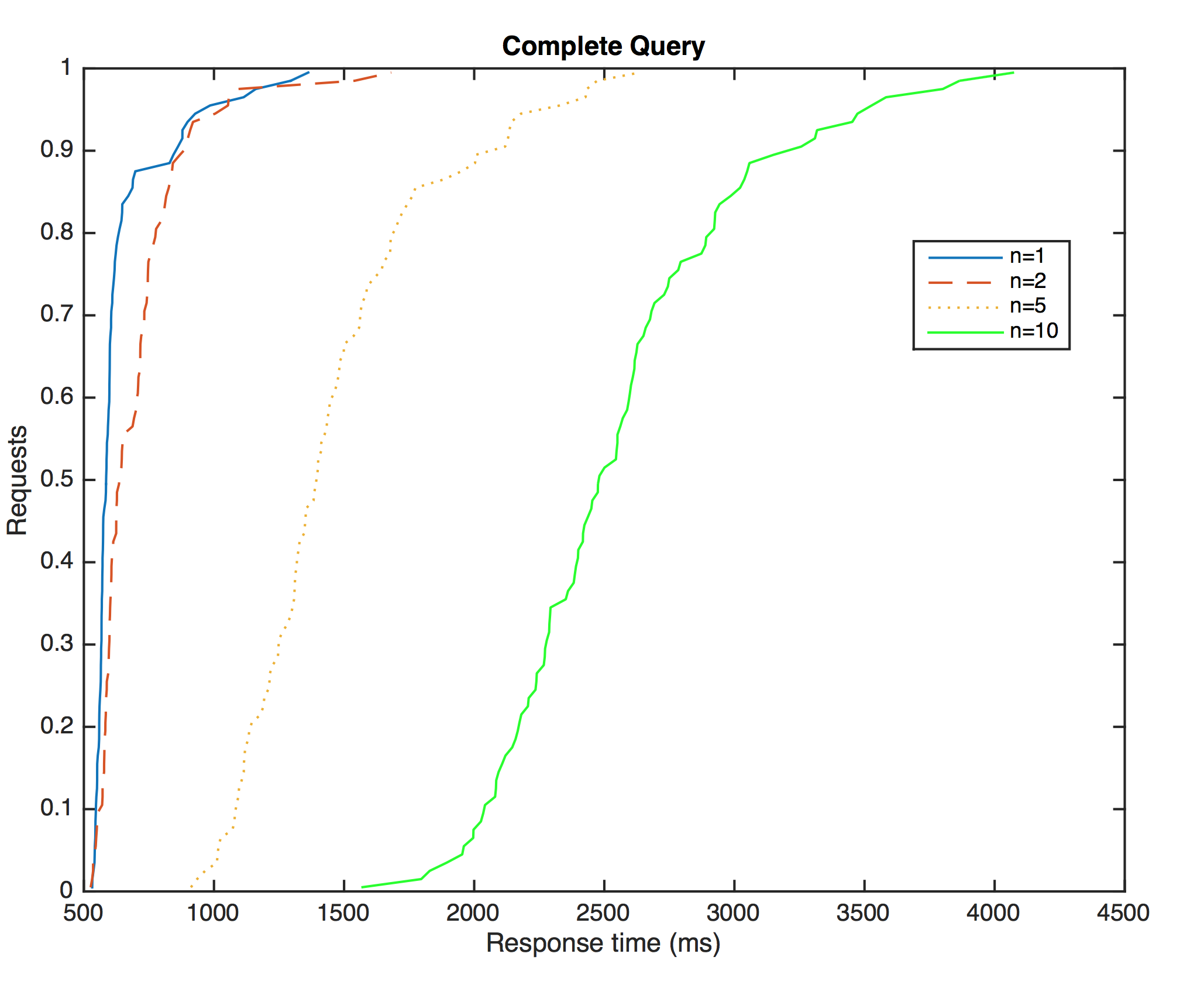
Discovery Service



Global Registry



Domain Registry



Complete Query

As expected, the higher the number of concurrent requests, the longer the reply time. We can also observe that the Domain Registry provides the fastest response. This was expected as this is the component that has to handle the highest number of queries and update requests. As such, it was designed to be highly performant. The Global Registry's response time is slightly higher as expected. This is due to the fact that the Global Registry has to contact several peers of the DHT in order to provide the reply. However, as the number of peers grows, the response time is expected to only grow logarithmically. The Discovery time takes the longest to respond and is the greatest contributor to the overall query time. This is partly due to the fact of this server being the most distant.

We can also observe that for all the services, the 80% fastest queries lay within a small time band, showing a very consistent response time. A greater variability is observable in the 10% slowest queries.