reTHINK framework evaluation through application development

Bernardo Graça
bernardo.graca@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2017

Abstract

With the goal to be an alternative to the currently dominant walled garden communication networks, the reTHINK project provides a new framework for web application development that handles with governance, security and identity management for the registered users. In addition, this European project aims to offer a global, open and shared identity system enabling dynamic trusted relationships among distributed applications without relying on communication protocols. This document presents an evaluation methodology with the goal of evaluate the reTHINK project from the point of view of ease in developing applications when compared with traditional technologies, testing the benefits and costs in terms of complexity. To accomplish this goal, we developed two versions of the same web application and performed an evaluation considering three different points of view. With this work, we showed the main advantages and disadvantages of using the reTHINK framework for application development, justifying them and providing alternatives to those that need to be improved. At the end and considering all the conclusions, we provide the most important recommendations to make the reTHINK framework more usable, easy to use and, consequently, more well accepted and adopted by the developer community.

Keywords: framework, application, reTHINK, web, users, development

1. Introduction

The telecommunications industry is evolving at a dizzying rate. In fact, today, the traditional telecommunications based services, like voice telephony, are losing importance at a fast pace. In addition, users are looking for solutions over the Internet that bypasses the traditional operator's distribution.

To fill the demand, the Over-The-Top (OTT) services started to grow drastically, especially after the introduction of the Web Real-Time Communication (WebRTC) standard [1] that enables real-time communication capabilities between browsers for audio, video and data exchange over Peer to Peer (P2P) connections. This type of services changed not only the way of communication between users but also the way to access the media through the Internet without any kind of involvement from the user's Internet Service Provider (SP).

On the other hand, most of these services and applications, operate on a closed system, also known as walled garden [2]. Nowadays, we have Skype from Microsoft, Facetime from Apple, and Duo from Google. Each big company has its own equivalent service, each stuck in its own bubble. These services may be great, but can be problematic since they cause vendor lock-in and limit the portability of user identity and data [3]. In addition, we have the social factor. Users have friends that uses all types of applications and services. To interact with them, users can not use their favorite application or service because they are not interoperable. Instead, users are forced to use the same applications, making it extremely difficult for new applications to succeed, on a competitive and already crowded market.

A project that is trying to be an alternative to the currently dominant walled garden communication networks, is the European founded reTHINK[1] project that describes a new framework for application development.

The reTHINK project gives a greater emphasis to communications, providing solutions to manage real-time communication capabilities, aiming to create dynamic trusted relationships between distributed applications and implicitly interoperable. To achieve such goal, the reTHINK project involves the creation of dynamic web-based services named Hyperties, which remain active for the duration of the particular service logic that has instigated their creation. In addition, it leverages the Protocol-On-The-Fly (ProtoFly) [4] concept to avoid the creation

[1] https://reTHINK-project.eu project, last accessed July 2nd, 2017

or modification of the standard network protocols, enabling the communication between different Hyperties from different SPs.

The motivation behind this work was the emergence of the reTHINK project. Programmers are encouraged to developed new communication enabled applications using the reTHINK framework. As a consequence, this will allow users from different reTHINK enabled applications to communicate with each other without the need to use the same standard network protocols. The reTHINK project is in the final stages of development, but a functional and stable version is already available that can be used by programmers to develop their own applications. At this stage, an evaluation from different points of view over the reTHINK project is needed. Specially, from the point of view of ease in developing web applications when compared to traditional technologies, testing the benefits and costs in terms of complexity.

2. Background

Usually, a web application is a client-server software in which the client or User Interface (UI) runs in a browser. Common web applications include webmail, online retail sales, online auctions, wikis, instant messaging services and many others. To implement these services correctly, the developer should consider some important aspects.

2.1. Web Application Development

The technologies used for web application development are an important decision that every programmer needs to make. There are two main categories of coding, scripting and programming for creating web applications: client side and server side.

In the client side, the code is executed or interpreted by browsers and it is generally viewable by any visitor. There are some common technologies used for client side coding, scripting and programming such as Hypertext Markup Language (HTML), CSS, JavaScript and jQuery. On the other hand, in the server side, the code is executed or interpreted by the web server and is not viewable by any visitor. There are some common technologies used for server side coding, scripting and programming such as PHP, Ruby, Python, JavaScript and Java.

2.2. Communication

One important aspect that the programmer should consider when dealing with web applications is the communication protocol used to exchange data, generally between the client (browser) and the web server. Web applications use Hypertext Transfer Protocol (HTTP) that already provides communication capabilities to the applications using the request-response paradigm [5]. However, the programmer has available other protocols based on different rules and specifications such as Remote Procedure Call (RPC), WebSocket and WebRTC.

2.3. Security in Web Development

Another very important aspect that should be considered is security. The developer needs to write code that fulfills customer functional requirements. And it needs to be fast. Somewhere, way down at the bottom of the list of requirements, behind, fast, cheap and flexible is "secure". That is, until something goes wrong, until the system built is compromised, then suddenly security is the most important thing.

Web application development considers many security considerations such as data entry error checking through forms, filtering output, cryptography, secure communications and sandboxing mechanisms.

2.4. Web based Identification and Authentication

The identification and authentication of users is also another very important aspect in web application development. In general, web applications use the HTTP protocol, which is a stateless protocol. For this reason, the web server does not retain any information or status about each communication partner across multiple requests. However, for most of the web applications it is almost mandatory to know who connects to the server, for tracking an application usage and to deliver the dynamic content requested.

To solve this problem, the applications needs to implement some mechanisms that provide two important features: the association of a statement to the communication partner (identification), and that proves the identity that the user says to be his (authentication). Examples of these mechanisms are cookies, HTTP authentication and OAuth protocol.

3. Related Work

In this current day and age, pushing out a finished and polished application well before a competitor is key. This is where web development frameworks come in. A framework can be seen as a base or a skeleton to build upon, allowing to speed up development by not having to rewrite features and structures that are commonly used in most web applications. Consequently, choosing a framework for application development is a very important decision. The developer should look for a framework that is appropriate for the type of application to develop and uses the desired programming language.

Usually, there are two classifications of frameworks for web application development: full-stack framework and non-full-stack framework. A framework is considered a full-stack framework if it helps

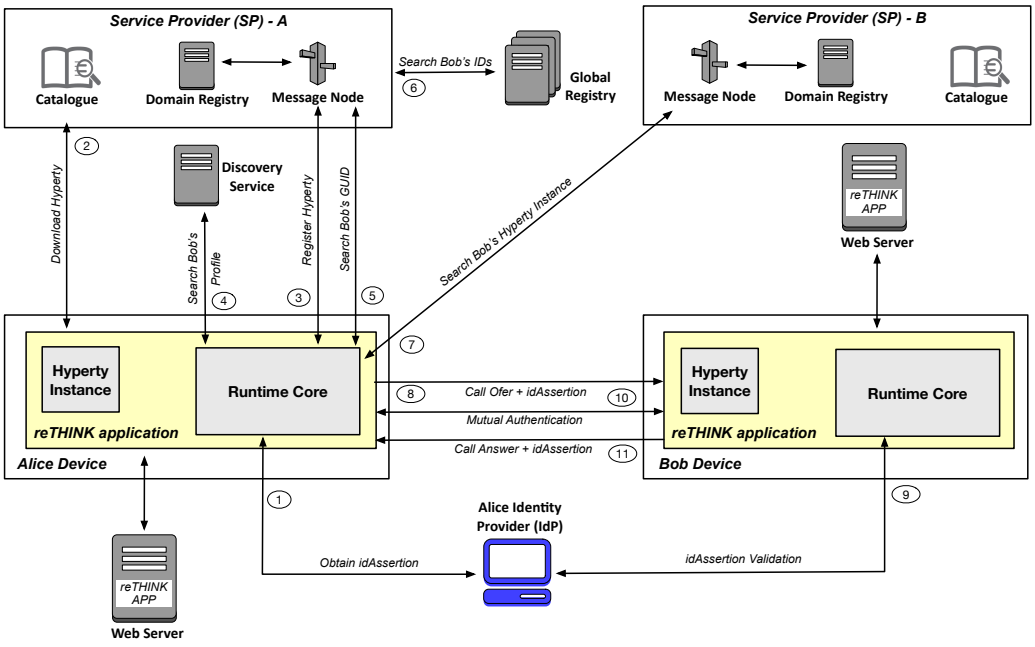

Figure 1: reTHINK framework usage example.

the developer with the full development stack from the UI (client side) till the data store (server side). If that does not happen, then the framework is considered a non-full-stack framework. Generally, a non-full-stack framework has a specific focus such as deal with the WebRTC standard or help with the UI development.

3.1. reTHINK Framework

The reTHINK framework incorporates two software concepts: the ProtoFly and Hyperty. The Hyperty is a new web service that can be described as a module of software dynamically deployed in web runtime environments on end-user devices, through simple, but sophisticated identity management techniques. These Hyperties are maintained by the SPs and their catalogue services helps the user to choose and download the most appropriate Hyperty instance. The communication between Hyperties is based on the ProtoFly concept that pulls code on-demand to dynamically select, load and instantiate the most appropriate protocol stack during runtime.

The reTHINK framework architecture can be described by four main components. Starting with the SP, which is comprised by several internal components such as Domain Registry, Message Node and Catalogue. The Domain Registry has the responsibility to register publicly the Hyperties upon their installation in the user device. The Message Node is the component responsible for intercepting and forwarding messages to be exchanged between users. And, the Catalogue is a repository that hosts the web oriented software to be used by user devices.

Then, the user devices are responsible for running the reTHINK client side application, composed by one or more Hyperties and the Runtime Core, both download from the SP Catalogue. The Global Registry and the Discovery Service components provides means for searching users using different types of information such as profile, interests and Global User Identifier (GUID).

To give an example of a complete operation of the reTHINK framework, it is demonstrated a use case where Alice initializes the reTHINK application from the start and then try to contact Bob, also running the reTHINK application. Figure 1 demonstrates the flow with the messages that need to be exchanged to achieve this objective.

When Alice stars the reTHINK application, the Runtime will be installed and the selected Hyperties will be loaded into the application. To successfully deploy an Hyperty, Alice needs to select an identity from the preferred Identity Provider (IdP) that will verify Alice's credentials and retrieve the idAssertion (step 1 of Figure 1). If valid, it starts downloading and installing the Hyperty from the Catalogue of SP A (step 2 of Figure 1). Once the installation is done, an Hyperty instance Uniform Resource Locator (URL) is generated and the Runtime Core associates Alice's identity with the Hyperty instance URL to register it in the Domain Registry of the SP A (step 3 of Figure 1).

To establish a communication channel, Alice needs to discover Bob's Hyperties instances (step 4 to 7 of Figure 1). Firstly, Alice uses the Discovery Service to search Bob's profile with the aim of retrieving his GUID. Assuming that Bob has a

profile, a query to the Global Registry is issued to resolve Bob's GUID into his IDs and SP domains. With Bob's IDs, it is now possible to query the Domain Registry of the Bob's SP that will reply to Alice's device with Bob's Hyperty instance URL, for which he can be contacted.

Finally, steps 8 to 11 represents the call initiation between Alice and Bob. The call initiated by Alice contains her idAssertion that will be validated by Bob's Runtime Core through the Alice's IdP. After validation, Bob's Runtime Core initiates a mutual authentication process to authenticate themselves mutually and to exchange the keys used to establish a secure communication channel.

4. Evaluation Methodology

To evaluate the reTHINK framework from the point of view of ease in developing applications when compared to traditional technologies, testing the benefits and the costs in terms of complexity, we defined an evaluation methodology divided into four parts.

Firstly, we defined a web application capable to evaluate the reTHINK framework by covering the most important features and use cases targeted by the framework. The defined application was a video conference application with chat and video functionalities. Users can use this application to create chat rooms to communicate with others through messages. In addition, users can also make video or audio calls belonging or not to a chat room. The authentication of users is done using their preferred IdP and after authentication, each user has access to their profile that it is created using the information (e.g. name and email) retrieved from the IdP used.

To be able to compare the development using the reTHINK framework with the development using traditional technologies, we proposed the implementation of two versions of the same defined application. One version was implemented taking advantage only of the features provided by the reTHINK framework, while the other version was developed using traditional technologies e.g. frameworks, libraries, databases, that a typical programmer would use to develop this type of web application. Along each implementation, we tried to understand what are the advantages, disadvantages and limitations of each used technology.

Once the implementation was complete, we applied an evaluation composed by three different perspectives. Firstly, we did an evaluation focused on the development of each application using different metrics such as documentation, programming effort and development time. The aim with this first evaluation was to gather the main advantages and disadvantages that a developer gets by using such technologies, especially with the reTHINK framework. Additionally, it was also important to compare the ease of developing applications with the used technologies.

After that, we proceed with an evaluation over the developed applications using two different types of tests. First, we did some performance tests to collect the efficiency of the most important features present in both develop applications, e.g. exchange of messages. It was important to test those features because they are directly related with the used technologies, allowing to gather its advantages and disadvantages. Afterwards, we did some portability tests to see the behavior of both developed applications in different environments, in this case in different browsers.

The last evaluation carried out was the users and developer's evaluation that makes this whole evaluation more robust and not so subjective. The user's evaluation was focused on the developed applications using a survey to collect some initial feedback about particular aspects detected along the implementation and evaluation of both applications. While the developer's evaluation was a much more complex evaluation, where experienced developers had the opportunity to use the reTHINK framework to develop their own web applications. In this last evaluation was used several code challenges and a huge survey with the aim of gathering feedback about the reTHINK framework considering their experience in this area. In both evaluations was conducted a statistical analysis over the answers to the questions, allowing to reach certain conclusions, specially about the reTHINK framework.

At the end and considering the results of this evaluation, we presented an overall analysis and some recommendations to the reTHINK project. We think that following these recommendations will make the system more usable and efficiency in a near future.

5. Video Conference Application supported by the reTHINK framework

To implement all the requirements of the application supported by the reTHINK framework, we tried to take advantage only of the features provided by the reTHINK framework and its components (see Figure 2). However, the view (UI) is out of scope of the reTHINK framework, so we decided to use an additional framework called Bootstrap[2]. This framework is the most complete and has dozens of customs HTML and CSS components that allow a rapid UI development.

5.1. Authentication of Users

This application uses two different Hyperties. One to deal with the chat rooms and exchange of messages called Group Chat Manager and another to deal with the video calls between users called Con-

[2]http://getbootstrap.com/ , last accessed July 17th, 2017

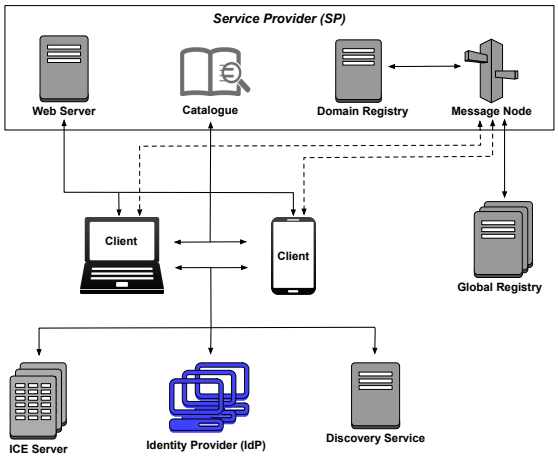

Figure 2: Architecture supported by the reTHINK framework.

nector. When the application starts to deploy these Hyperties, the authentication form provided by the reTHINK framework is automatically prompted and shown to the user. After authentication, the application should be able to obtain the identity object to fulfill the user profile. In this case, the application can obtain the identity object through the deployed Hyperty.

This application requirement was implemented using only and exclusively the features provided by the reTHINK framework, which allowed speed up the development with a low programming effort. However, this authentication system has some limitations that may not appeal to developers such as a hardcoded authentication form and no support for the creation of local accounts and anonymous identities.

5.2. Chat Rooms

The Group Chat Manager Hyperty is responsible to handle with the creation of chat rooms, invitation of users, exchange of messages between users and the join of users in the chat rooms. For this application, a chat room can be seen as a Data Object that is associated with the deployed Hyperty instance. The Hyperties communicate and cooperate with each other through a data synchronization model called Reporter - Observer (Figure 3). This model uses a P2P data stream synchronization solution for programmatic objects.

Figure 3: Reporter - Observer model.

To successfully create a chat room, the user can associate a name and the email addresses of the users that he wants to invite to the chat room. Through the Discovery component, the Group Chat Manager Hyperty can discover the registered Hyperties associated with those emails and invite them to join the chat room. If they accept the invitation, their Group Chat Manager Hyperty instance will subscribe to the Data Object (chat room) and act as an Observer.

To allow use cases like the exchange of messages between users of the same chat room, where the Observers need to have writing permissions, the reTHINK framework introduces the Child Data Objects. So, when a user sends a message, a Child Data Object is created in the Data Object associated with the chat room. This creation is consequently propagated to all the participants (Reporter and Observers) of the chat room.

These application requirements were also implemented using only the features provided by the reTHINK framework. However, unlike the previous requirement, these required a tremendous programming effort, especially at the Hyperty level. Firstly, the documentation is not well structured in order to facilitate the understanding of these complex concepts, e.g. Data Objects and Child Data Objects. And then, it is not very clear what libraries are needed and what methods should we use to achieve quantified goals.

5.3. Video Calls

The Connector Hyperty has the responsibility to deal with the establishment of video calls between users. This Hyperty, as suggested by the reTHINK documentation, uses the WebRTC Adapter.js[3] that provides the all-in-one WebRTC solution cross-browser. The Adapter.js is an open source project that enables web browsers with real-time communications capabilities via simple JavaScript Application Programming Interfaces (APIs).

To start a video call between two users, the discovery process needs to take place in order to find the Hyperty instance associated with the email address of the invited user. After that a Data Object is created following the same principles mentioned in the previous section. Like the chat room feature, the invited user has the ability to accept or reject the call. If the invited user accepts the call, they will share a media stream of P2P data (video and audio) with each other and the call takes place.

This last application requirement was implemented using only the reTHINK framework and the Adapter.js. Although, its implementation was quite straightforward, we think that required more effort than it would require with different technologies.

[3]https://github.com/webrtc/adapter , last accessed August 15th, 2017

6. Video Conference Application supported by the State of the Art

The second version of the proposed application was developed using traditional technologies with the aim of having a term of comparison for this evaluation. To implement the same functional requirements, we had to use several technologies (see Figure 4). For example, we choose the PeerJS framework[4] to handle with the signaling process and allow the transfer of P2P data streams (messages and video). For the authentication of users using their preferred IdP, we choose to use the Passport.js[5] because it can be unobtrusively dropped into any Express-based web application, provides several authentication protocols and works with several IdPs such as Google and Facebook.

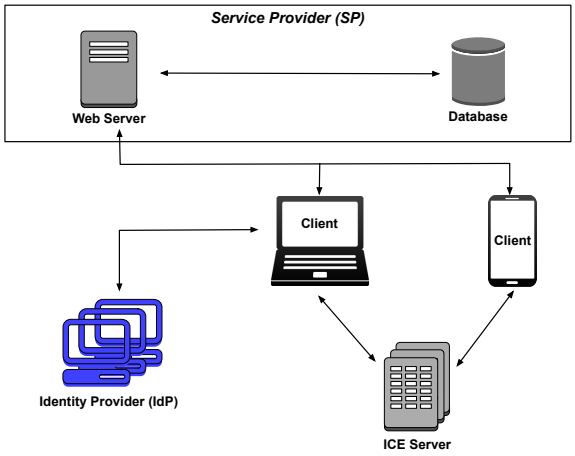

Figure 4: Architecture supported by the State of the Art.

6.1. Authentication of Users

With Passport.js, the authentication of users is just a matter of loading the appropriate authentication strategy and to configure the routes, callbacks and developer keys in the web server. In addition, and contrary to the application supported by the reTHINK framework, we had to develop from scratch the authentication form that is prompted and shown to the users. After authentication, we can retrieve from each IdP all the information necessary to form the user profile.

With this application, we wanted to go further and show how easy it is to add new authentication schemas through the creation of local accounts. To implement this additional requirement, we use a strategy provided by the Passport.js and the Bcrypt[6] library to ensure higher security in the storage of passwords and its validation.

Although the implementation of this authentication system was straightforward, it required a lot more programming effort than the required for the application supported by the reTHINK framework. In this application, we had to develop the web server from scratch, an authentication form and integrate a group of APIs. However, there are a few advantages that should be highlighted such as the ease in adding new authentication schemas or new IdPs.

6.2. Chat Rooms

The main goal of the chat rooms is to enable the exchange of messages between users. This ability is possible through the use of PeerJS framework. In our implementation, the web server is responsible for generating the peer identifiers essential to establish communication channels between users.

In this application, a chat room is a simple JavaScript Object Notation (JSON) object containing its name, URL and owner email. Every time, a user wants to initiate a chat room, a request to the server is sent through the Socket.IO[7] library for generating a unique URL to allow the join of other users.

The chat one-to-one user was quite simple to implement, but when we started the implementation of multi chat feature (many users), we had some problems. After reading the PeerJS documentation, we did not find any solution or suggestion for the implementation of this feature. With the aim to find some useful answers, we discussed this question on their official GitHub repository and we encountered one possible solution. The peer, owner of the chat room, will be responsible to broadcast to his all active connections, all the events and messages received (see Figure **??**). Although, we recognized that this solution should not be ideal because it creates some inconsistencies among peers, it allowed to achieve the implementation of this requirement.

The well-structured documentation of PeerJS framework allowed to implement this requirement with a much smaller programming effort. Although, we had to developed several components from scratch, design solutions to their interactions and integrate additional libraries to deal with several aspects, the final result was very good.

6.3. Video Calls

With the peer object configured, the client can start a video call whenever he wants, it's just a matter of dealing with the corresponding events. In this case, a call event is issued every time a user starts a video call. The invited user has the ability to accept or reject the call offer. If he accepts the invitation, the call proceeds and they will share a P2P data stream with each other. In addition, there is no need to

[4] http://peerjs.com , last accessed July 13th, 2017

[5] http://passportjs.org/ , last accessed July 11th, 2017

[6] https://www.npmjs.com/package/bcrypt , last accessed August 17th, 2017

[7] https://socket.io/ , last accessed August 17th, 2017

waste time with the signaling process or other issue because the active PeerJS server handles with all these questions.

The implementation of this last application requirement was very simple with a lot less programming effort than the required for the application supported by the reTHINK framework. At this stage, we do not see any advantage of using the reTHINK framework to deal with such requirement. The API provided by the PeerJS framework is simple and intuitive and we do not had any problem on the implementation of this application requirement.

7. Evaluation and Comparison Analysis

The evaluation carried out in this work was divided into three parts (development, applications and, users and developers evaluation) that are described in the following sections. The aim of this evaluation was to evaluate the reTHINK framework from the point of view of ease in developing applications when compared with the traditional technologies, testing the benefits and the costs in terms of complexity.

At the end, we provide an overall analysis and a few recommendations considering all the conclusions taken about the reTHINK framework in each part of this evaluation.

7.1. Development Evaluation

In this part, we tried to gather the most relevant metrics to evaluate the used technologies from the point of view of ease in developing applications. We choose the documentation, programming effort, development time and, ease of installation and deployment as the most appropriate metrics for this part of the evaluation. All the data used in each metric was gathered from both developments of the video conference application.

The documentation is often key to the success of the framework or any other type of technology. For this metric, we measured the time that we spent reading the documentation of the used technologies in each application development. The application supported by the reTHINK framework required 12 hours of documentation reading, which is approximately 42% more time spent than with the other developed application. In our opinion, this is due to the unstructured documentation that the reTHINK framework provides, when compared with the documentation of other APIs.

The goal of using a framework for web application development is to reduce the programming effort by not having to write everything from scratch. This is an important metric for developers when they are selecting the development technologies. In both developed applications, we had to use and code the same number of files. However, the application supported by the reTHINK framework required more 20% code lines (see Table 1). We think that this difference is not relevant enough to be taken into account in the developer's decision. Most of it is due to the Hyperties development, where we had to program all the interactions with the reTHINK components using different libraries.

Technologies	Number of files	Code lines
reTHINK framework	12	2807
Traditional technologies	12	2022

Table 1: Programming effort overview.

The third metric used in this part of the evaluation was the development time. The development of the application supported by the reTHINK framework (28 hours in total) took longer than the development of the application supported by the State of the Art (23 hours in total). This situation occurs in practically all the application requirements.

However, the time spent in the development of the chat rooms and video calls in the application supported by the reTHINK framework was not the expected. We think that it was a consequence of the unstructured documentation provided by the reTHINK framework. Although, the reTHINK framework provides concrete examples to implement some functionalities (e.g. exchange messages), it would be very good if it were provided widespread examples such as how to create a Data Object and what APIs the developer should use to manage the created Data Object.

The last metric used in this part of the evaluation was the ease of installation and deployment of both developed applications. For this metric, we collected for each developed application the number of dependencies to install, steps until installation (commands), size (megabytes) and average installation time (seconds). In addition, we analyzed the ease of deployment for each developed application, namely the additional configurations.

The video conference application supported by the reTHINK framework takes more time and require more steps to install, its bigger and have much more dependencies than the other developed application. The steps until installation are a consequence of the number of components, all initialized separately with different tools. These steps could be improved a lot with the use of the same tool and initialization file for all the reTHINK components. The size is also worrying but natural given the 95 dependencies. After a quick look into it, we conclude that more than a quarter of the dependencies are for testing purposes. We think that with a revision, this number may be lower.

Finally, the deployment is also a very important aspect when dealing with web applications. The application supported by the State of the Art does not require any additional configuration. However, the deployment of the application supported by the reTHINK framework could require more effort and additional configurations.

7.2. Applications Evaluation

The next part of this evaluation is the applications evaluation. In this part, we used both developed applications to perform some performance and portability tests. All the developed tests are JavaScript scripts that run with the applications. We tried to execute them over the same conditions to avoid potential network issues and its results are always presented using the Cumulative Distribution Function (CDF). Considering the requirements of the developed applications, we defined three types of performance tests: deployment, user search and message delivery tests.

The aim of the deployment tests was to measure the deployment time of each application in the client browser. For the application supported by the reTHINK framework we did three different tests: Runtime installation test, Hyperties deployment test and web page loading test. On the other hand, the application supported by the State of the Art has no elements deployed at execution time with exception of the web page shown to the client. For this reason, with this application we only used the web page loading test. As expected and as a consequence of the installation at execution time, the use of the reTHINK framework has a huge impact in the deployment time. The deployment of the application supported by the reTHINK framework takes in average more 1830 ms (see Table 2). We think that this time is relevant enough to bother users and developers. To improve this deployment time, one possible solution would be to create a lighter version of the Runtime, excluding some components.

Technologies	Test	Average Time	Total Time Average
reTHINK framework	Runtime Installation	857 ms	2173 ms
	Hyperties Deployment (concurrently)	934 ms	
	Web Page Loading	382 ms	
Traditional Technologies	Web Page Loading	343 ms	343 ms

Table 2: Deployment tests overview.

The user search tests are designed to test the efficiency of the discovery of users with the discovery system provided by each developed application. In these tests, we gathered the time necessary to complete the discovery process in 100 executions. Additionally, we varied the number of concurrent requests, using 1, 2, 5, and 10, between each execution. Taking into account the Figure 5 and 6, we can conclude that the discovery system provided by the reTHINK framework have in all cases a better performance. In addition, the discovery system provided by the reTHINK framework provides higher security in the communication between components using Hypertext Transfer Protocol Secure (HTTPS) and Mutual Authentication protocol.

To complete the performance tests, we designed the message delivery tests. The aim of these final tests was to measure the time necessary to deliver a message to another client using both developed applications. In these tests, we fired 100 messages, one every 50 milliseconds, in both developed applications. The results show that the exchange of messages is faster in the application supported by the State of the Art. This was already expected because the Data Object mechanism involves a bigger message flow and events propagation between components, whereas in the application supported by the State of the Art, the data is sent directly to client browser via P2P. At the end, we think that the difference between results are not relevant enough to get noticed by users and developers.

For the portability tests, we tried both developed applications in different browsers. We choose to use the Google Chrome 56.0.2 and the Mozilla Firefox 55.0.2 because both should support the WebRTC technology. The application supported by the reTHINK framework worked better than the application supported by the State of the Art using different browsers. However, we found some bugs in both developed applications that are related with the used technologies. Most of these bugs appears because this type of technology is constantly changing, especially the browsers. A framework for web applications development needs to be aware of these changes so it can keep up with this evolution.

7.3. Users and Developers Evaluation

The last part and probably the most important part of this evaluation is the users and developers evaluation. At this point, the opinions of users and developers are extremely important for the reTHINK project.

Firstly, we started by doing tests with users using the developed applications. The aim of these tests was to gather some initial feedback about the reTHINK framework through the comparison of both applications. To be able to get a relevant feedback, we gathered 10 users (8 males and 2 females) to try

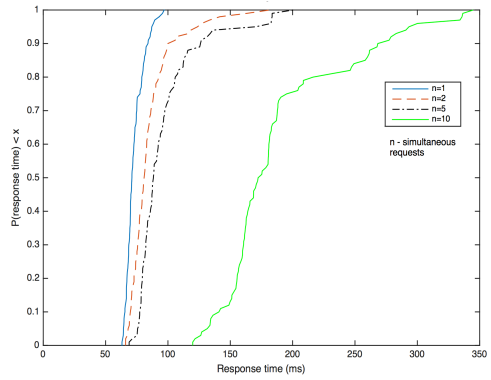

Figure 5: Application supported by the reTHINK framework.

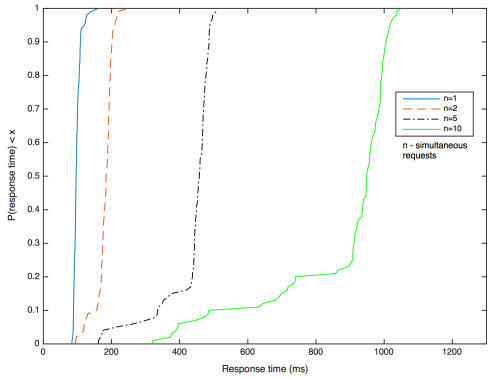

Figure 6: Application supported by the State of the Art.

both developed applications. Only 20% of the users have a background related with computing science, which means that most of them were not familiar with application development and programming. In order to collect useful feedback, we asked the users to load and authenticate themselves in both applications. In addition, we gave them complete freedom to use the chat room and video calls features. After having used both applications, we presented a survey to gather the desired user feedback. Considering all the collected feedback, the following conclusions are highlighted:

- Most of the users stated that the application supported by the reTHINK framework takes too long to load;
- The reTHINK framework should allow the development of the authentication form or the customization of the default provided;
- The identity management features provided by the reTHINK framework needs some improvements;
- The difference of efficiency in the search and messaging features between both applications are not relevant enough for the users.

On 27th of May, the INESC-ID together with AlticeLabs organized a developer competition[8], which took place in the Techtris House, a startup incubator and home to the WebSummit offices in Lisbon. The goals of this competition were to promote the reTHINK framework and gather the intended developer feedback. The competition had developers, who were completely unfamiliar with the reTHINK framework, to complete a set of 5 code challenges. The competition had 12 participants, grouped into 8 teams of at most two elements. Most participants were male, but a female attended. Participant's ages ranged from 20 to 70 years. The conclusions were reached using one survey that was used to evaluate different aspects of the reTHINK framework. This survey was divided into seven sections. One section for team characterization and background, five sections corresponding to each of the challenges and a last section for overall evaluation. The collected feedback from this survey will be used to identify and improve on the weaker parts of the framework, making the system more usable.

This event was a success, allowing us to collect very useful feedback. Considering all the collected feedback, the following conclusions are highlighted:

- Participants found Hyperties easily to integrate into applications. However, more examples are required;
- All participants considered the reTHINK framework easy to use;
- The reTHINK library logging needs improvements because it is too verbose;
- Documentation needs to be improved, more structured, generalized examples and demos;
- All participants expressed the intention to use the reTHINK framework again in future projects;
- It would be great if the reTHINK framework was supported by most browsers;
- The authentication form needs improvements in terms of UI;
- The Hyperty Toolkit was appreciated by participants;
- It would be great if reTHINK had built-in integration with front-end frameworks and examples.

[8] https://www.meetup.com/pt-BR/Decentralised-Comms-LX/events/239818330/?eventId=239818330 , last accessed August 27th, 2017

7.4. Overall Analysis and Recommendations

After analyzing this evaluation as a whole, we think that the ultimate goal of this thesis was achieved. We were able to evaluate the reTHINK framework from different perspectives, and, whenever possible, compare it with other technologies. Additionally, we had tested its advantages and disadvantages through the application development using several evaluation metrics.

This evaluation was extremely important for the reTHINK project because it allowed to identify the necessary improvements in order to make the system more usable and developer friendly in the future. Besides that, we gathered a good amount of users and developers feedback, which at this point it is very important for the project. In fact, some parts of this evaluation were used in the internal documents of the reTHINK project. Taking into account all the conclusions made at each part of this evaluation, we can provide a few recommendations to the reTHINK project.

The most important recommendation and probably, the most urgent is to improve their documentation. At this point, the reTHINK framework is ready to be used by developers, however, it is extremely difficult to understand its concepts and features just by reading its documentation. In our view, the reTHINK project should gather a team just to improve the documentation through the development of a website to expose better the information, making it more appealing and easy to understand.

The second recommendation is related with the size of the Runtime that each application must use. We think that its size is too big, not only affects the deployment time of the applications, but it also could cause the loss of interest of some developers. There are several components in the Runtime that are not used, so this problem could be quickly solved by creating a lighter version without these components.

To conclude, we think that it is also important to highlight the positive aspects collected with this evaluation. Most of the users and developers recognized the potential of the reTHINK framework and its uniqueness. They also enjoyed using the reTHINK framework to develop their applications and have shown interest in using it in the future. This shows that the reTHINK framework can be very well received by developers and it has a great margin of progression with a few improvements and integrations.

8. Conclusions

This document describes an evaluation methodology focused on the reTHINK project. At this point, this European project is almost finished and, consequently, it was necessary to collect an evaluation not only focused on the efficiency of its components.

In this in mind, we aimed to evaluate the reTHINK project from the point of view of ease in developing applications when compared with traditional technologies, testing the benefits and the costs in terms of complexity. The achievements of this evaluation, allowed us to provide important recommendations to the reTHINK project, so it can be more usable, easy to use and efficient in the future.

We know that this work has a high degree of subjectivity because most of the collected data derives from our programming skills, knowledge and the technologies that we choose. However, most of the companies when develop something new follows the same path and yet, most of them does not evaluate considering so many points of view. With this in mind and although we recognized its subjectivity, we think that we did what was necessary to be done in order to overcome the main goal of this thesis and present a solid evaluation and important recommendations to the reTHINK project.

References

[1] Adam Bergkvist and Daniel Burnett and Cullen Jennings and Anant Narayanan. WebRTC 1.0: Real-time Communication Between Browsers. *Working draft, W3C*, 2017.

[2] Tim Berners-Lee. Long Live the Web. *Scientific American*, 303(6):80–85, 2010.

[3] Harry Halpin. Beyond Walled Gardens: Open Standards for the Social Web. *SDOW2008, Karlsruhe*, 2008.

[4] Paulo Chainho and Kay Haensge and Steffen Druesedow and Michael Maruschke. Signalling-On-the-fly: SigOfly. In *Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on*, pages 1–8. IEEE, 2015.

[5] Roy Fielding and Jim Gettys and Jeffrey Mogul and Henrik Frystyk and Larry Masinter and Paul Leach and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical Report RFC 2616, Internet Engineering Task Force, 1999.