EPFL

Columbus United

A Byzcoin Explorer

Sophia Artioli & Lucas Trognon

Autumn Semester 2020

A Bachelor project hosted by the
Decentralized Distributed Systems Laboratory (DEDIS) at **EPFL**

Supervised by Noémien Kocher

Directed by Pr. Bryan Ford

Contents

1 Introduction

1.1 Motivations

Blockchains are a notoriously hard concept to understand, which makes them difficult to use for broader audiences. In particular, they are difficult to trust without a deep understanding of their inner workings which might require some sort of background. Blockchains are a powerful tool that is used amongst others for governance purposes or crypto-currencies. However, one must be able to easily verify their contents and check that properties are fulfilled to use and trust it. In this matter, Columbus enables a concrete visualization of Byzcoin, a blockchain created by the DEDIS lab at EPFL. The Columbus explorer displays the chain and diverse information about each block to help users have a better understanding of the underlying concepts.

1.2 Byzcoin

ByzCoin is a consensus protocol aimed at delivering a high processing speed and immutable validations to Bitcoin transactions while being Byzantine fault robust. The use of communication trees to optimize transaction commitments and verification enables it to achieve a transaction latency of fewer than 30 seconds. Which reduces the possibilities of double-spending and selfish mining. This is done by creating collective groups of miners to work on validating transactions. Inspired by protocols such as the Practical Byzantine Fault Tolerance [13], miners now only require the validation of two-thirds of the other members of their group for each transaction to be processed.

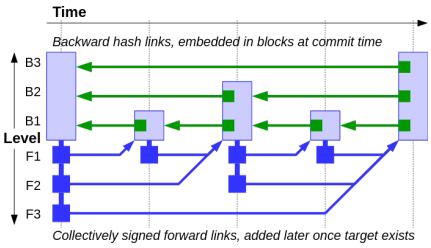

Figure 1: Diagram of a few Skipchain blocks

Byzcoin is contract-based. Each contract is called an instance, and can be Spawned (created), Invoked (modified), or Deleted (self-explanatory) by the users through the use of instructions to update the ledger. Each instruction is carried to the nodes by a transaction, and these transactions are saved in the blocks. Byzcoin is built using the Cothority framework developed by the DEDIS lab. The Cothority framework handles the communication between several nodes and allows them to construct a collective authority (thus the name) consisting of conodes. Conodes form a roster to sign and verify processes on the Skipchain. The Byzcoin ledger's underlying blockchain is called a Skipchain [2, 4]. The particularity of Skipchains is that blocks are assigned a height based on their indexes (in the Byzcoin implementation, the height

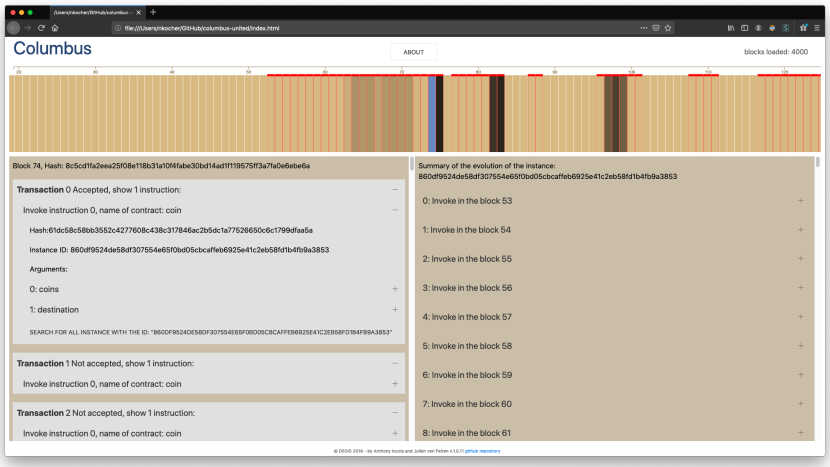

Figure 2: Columbus v0.0.2 main screen

equals the number of times the index can be divided by 4). Every block of height k will link to the nearest block of size k, the nearest block of size k-1, etc... (Figure 1) This architecture allows fast browsing of the chain by lessening the number of iterations required to find a block.

1.3 Columbus' history

Columbus [15] started in March 2020 as a bachelor project. Julian von Felten and Anthony Iozzia were tasked with developing a tool centralizing all the features of the ByzCoin explorers that already existed, under the supervision of Noémien Kocher. v0.0.1 was then released in June 2020. Noémien Kocher updated the documentation, squashed bugs, and expanded the documentation to produce v0.0.2 at the end of August (Figure 2), shortly before we started working on Columbus. We reused as much previously accomplished work to continue working in the same general direction. On the way, we added modifications and created new features.

2 UX feedback driven-workflow

As stated above, the foundation of a working interface was already present. What we believe were the main issues in the previous explorer were the lack of polish on the user experience and the fact that some features seemed incomplete. To refine it as much as possible, we constructed a list of features to implement around user feedback we received. This helped us assess the work that needed to be done and organize ourselves throughout the semester.

2.1 Goals

First, we used a list of high-level user experience scenarios that Noémien Kocher provided us to define what the users should be able to accomplish with the UI.

- Advanced users (the people working on Byzcoin) should be able to understand the interface and make sense of each field by themselves. They should not need to read a tutorial to find all the features.
- Intermediate users (people that have basic knowledge of blockchains) should be able to browse the Skipchain without being too confused and getting a better grasp at how Byzcoin works. Given a few pointers, they should be able to understand most features.
- Novice users (people with no knowledge of blockchains) should be able to broadly understand the concept of a Skipchain and to get a grasp of the main information displayed.

2.2 Designing the first round of UX

We designed a protocol and questioned 10 users on their experience using the explorer: We tested what the users were able to do, how they did it, and why they couldn't do more.
First, we started by mentioning that they were about to interact with a blockchain explorer. However, we did not give any more information. The user then proceeded intuitively. The guidance was given as soon as the user did not know how to interact further with the platform. Users were given tips such as the zoom and scroll features of the chain, clicking on blocks of different colours or being able to get more information about transactions. Finally, the user answered these questions:

- Does the design naturally remind you of blockchains and/or crypto-currencies?
- What given notions do you understand?
- Is the information relevant for your use of the explorer and what is missing?
- Did you notice that this blockchain is a Skipchain?
- Do you know what a Skipchain is?
- What features (i.e. zoom/scroll) did you intuitively use?
- Did you notice the about button?

- How did you find the arrangement of the page? (colours, optimization of space, etc)
- What was your global experience and what knowledge did you acquire from the explorer?

2.3 Feedback from the users

Novice users were usually confused and lost, they did not know how to interact with the page nor what the displayed information meant. Additionally, they did not realize the displayed chain is a Skipchain and did not understand any related information.
Intermediate users were struggling as well while it seemed that no one in the lab really used the explorer. They understood the basic concepts found in other blockchains but did not understand Byzcoin nor Skipchain related elements.
For both of these categories, users did not realize that the displayed chain wasn't a regular blockchain, they particularly questioned the concepts of forward and backward links.
Expert users needed more on-hand information and more flexibility moving around the chain, additionally the inner working of Byzcoin did not seem displayed on the initial explorer.
Aesthetically speaking, all categories of users generally raised the same points:

- The explorer is opaque. Users were aware that information was there but they did not know where and how to get it.
- Users were unsure of what they had to do once the page was loaded.
- The space occupancy was not spread equally throughout the explorer. Some spaces were left blank and others were full of information.
- The colour scheme was neither aesthetic nor eye-catching.

We realized that we had to work on two different key usability concepts to push Columbus further. On one hand, we needed to make the tool more accessible to neophytes while providing more advanced features to expert users. We had to focus on both learnability and the efficiency of Columbus.

2.4 Assessment of the features needed

From the received feedback, we assessed what were the most important and urgent features to be implemented:

- A search bar, as navigating solely by clicking on blocks is not efficient. The chain takes a certain amount of time to load so going from one part of the chain to another takes a significant amount of time.
- A dedicated space displaying the last validated block and its characteristics.
- A proper visualization of the Skipchain. Simple rectangles are not evocative enough, the height and links of the blocks need to be shown.
- An overhaul of the instruction argument section.
- A redesign of the UI to make it more appealing and easy to understand.
- A way to hide hashes to reduce the clutter of the interface.
- A more comprehensive list of block details.
- A better way of displaying the life cycle of an instance.
- A way to help the user understand the purpose of each field.

The main focus was to help all sorts of users achieve their goals on the explorer. From these features, we designed a first sketch (Figure 3) of what we felt Columbus would or should look like. Initially, the explorer was really opaque and quite difficult to use even for more advanced users. We realized that information needed to be more accessible and easier to read. Additionally, we felt that the layout of the page needed to be reformatted and space better optimized. We also emphasized the fact that the displayed chain was a Skipchain since very few users realized that.

Figure 3: First sketch of the Columbus 3 UI

2.5 Workflow

Stack : From a technical point of view, we used the following stack to implement new features to the explorer:

- Typescript, as the front-end language
- NPM, as the package manager
- Webpack, as the bundler
- RxJS, as the reactive programming library

- Webpack, as the bundler
- UIkit, as the CSS framework

List of features : We defined a list of features, which combined would allow the UI to reach a result closed to our planned sketches. Once they were sorted by priority, we divided the work into two. Sophia would work on the features that were related to the chain visualization and the search bar, while Lucas would focus on general changes, instance tracker, and the block details part. To keep track of our progress, we used a GitHub project with a ticket for each feature. [14]

Road map : Starting the project, we defined steps and deadlines during the semester to get to the new explorer we had imagined. The following timeline was mainly for reference, but it helped us stay synchronized throughout the project (Figure 4)

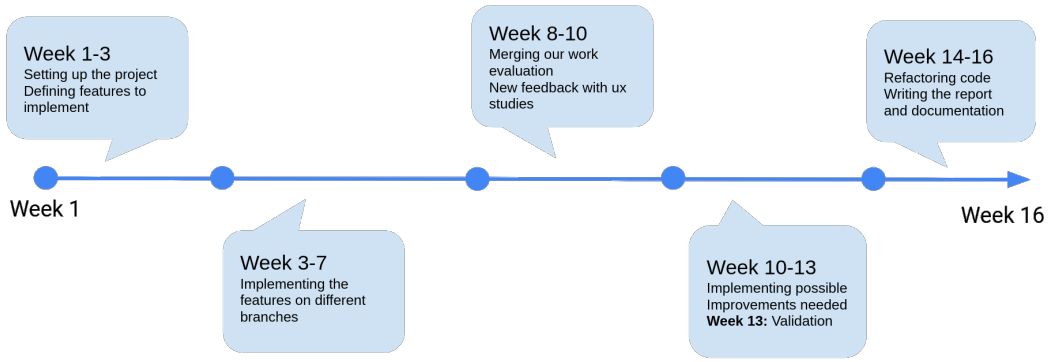

Figure 4: Time line of Columbus III developement

3 First round of implementation

3.1 General changes

3.1.1 Color change

In order to kick-start the project, we first went for a color change. The changes were mostly aesthetic. Although we noticed that the bold blue and white background we used was more contrasting than the previous colours. This helped us to better control the gaze of the users. Important information was made more visible, and the clutter was less oppressive. We chose two shades of blue.

The lighter shade is used for anything that is linked to the user selection, while the darker shade is used for everything else.

Figure 5: Old color scheme

Figure 6: New color scheme

3.1.2 Adjusting the axis scaling

The axis in place would not follow the transformations of the chain correctly. When the chain was un-zoomed, blocks and axis' ticks were not aligned, which was confusing. The user could not properly visualize where they were situated on the chain. This issue is now fixed, and the axis and block alignment is accurate.

3.1.3 Dynamic URL (used to revert changes or share an URL)

With all the new interactions that we provided, it was easy for new users to alter their selection in a way they didn't anticipate. In order to relieve some of the stress of making an action that is cumbersome to revert, users can now easily backpedal by using the "back arrow" integrated with their browser. It also enables a more efficient way of sharing the details of a block for debugging purposes for example.

3.1.4 Blockies

We insisted on having almost no visible hashes on the page. We believe that hashes are computer language, not human language, so it made no sense to display them. Instead, we use the hashes as a seed to generate a small 32x32px image called a blocky (Figure 7). As they vaguely resemble a face (because of their symmetry), they enable the user to easily recognize and compare instances or users.

Figure 7: Example of blockies found in the explorer

3.1.5 Tooltips and copy to clipboard

We got the hashes "out of the page" with the blockies and other various features. But we still needed them as they are valuable when doing searches, or for more advanced users. In order to make them available without cluttering the space. We made it so the hashes are initially not visible, but when hovering

a blocky for instance, a tool-tip shows up, displaying the hash associated with the blocky. If the user wants to use that hash elsewhere, he can simply click on it to copy it to their clipboard.

3.1.6 Flash messages upgrade

Flash messages gave the impression to some users that they did something wrong, and that they were cluttering the space a lot. We reformatted most flash messages so they would be less threatening and made them disappear after some time.

3.1.7 Automated versioning

When developing, we wanted to test our interface on a live version to make sure it was running correctly. It was also really useful to organize our UX studies. To make it less of a struggle when updating a new version, we made a small script when deploying that prevents browser from using outdated cached scripts.

3.2 Chain related implementations

The upper part of the explorer is dedicated to chain visualization. We modified the latter to have more on-hand information when initially landing on the page. In this way, the user already has some knowledge about the chain without having to do a single click.

3.2.1 Height Visualization

One main characteristic of Skipchains is that a block's description contains a height argument. In this sense, blocks have been reformatted in order to be displayed on the chain by their heights instead of being square blocks. The previous visualization was very opaque, the fact that blocks are of different heights offers more visibility and a lighter aesthetic (Figure 8)

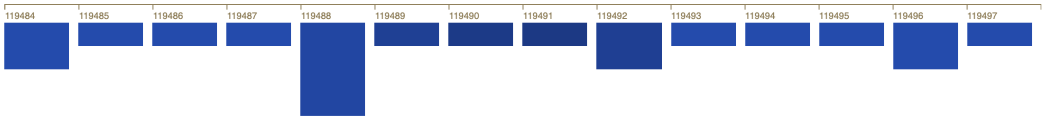

Figure 8: Part of the chain with blocks of different heights

3.2.2 Forward and backward links visualization

Another characteristic of Skipchains is that blocks are associated with forward and backward links. Therefore, arrows have been added between blocks that are connected in order to have a better visualization of how blocks are linked throughout the chain. This feature combined with the blocks displayed by their heights offers concrete and lighter visualization of the Skipchain.

3.2.3 Translation

The explorer did not offer any flexibility moving around the chain i.e. jumping to blocks that are not close index-wise. Therefore, we have implemented a translation feature to enable easy navigation throughout the blocks of the chain. This has been particularly helpful when implementing the search bar. For this feature, a new class called `Chunk`has been created. Chunks represent autonomous parts of the chain that have already been loaded and where blocks have been displayed (Figure 9). When the chain translates to new coordinates new bounds are calculated according to the transformation of the chain. If these bounds intersect with an already existing chunk, blocks do not need to be fetched from the client. On the other hand, if the bounds do not overlap with any of the existing chunks, a new `Chunk`is created and added to the list of existing chunks, along with its bounds. Then, we fetch the blocks from the client and display them for that specific part of the chain. This allows us to not reload all blocks when scrolling to different coordinates.

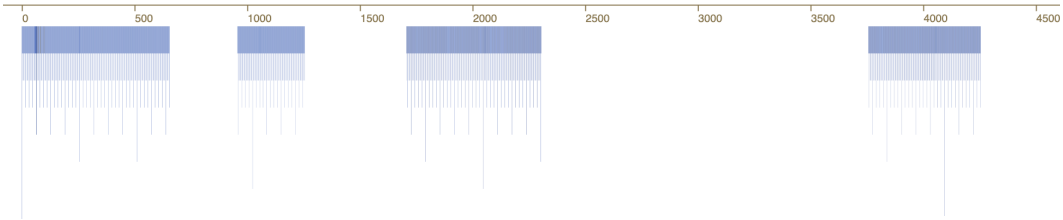

Figure 9: Four autonomous parts ("chunks") of the chain

3.2.4 Navigation with the forward arrows

Now that arrows are displayed and that we can move throughout the chain easily, the arrows between blocks have been made clickable. When a user clicks hovers on an arrow, it is highlighted in a different color. When a user clicks on it, the chain translates itself to where the arrow is pointing and displays the information about that block.

3.2.5 Last Added block

Space has been dedicated to the visualization of the last added block of the chain. In this matter, the user could be anywhere on the chain and still access information about the last added block easily. The fact that the last forward links of the chain are broken, helps the user know the real length of the chain since this cannot be displayed at the moment. An additional point that was raised by the user experience studies was that users did not know what they were expected to do on the load of the page. For this reason, an animation has been added. The last added block of the chain slides next to the chain and its information is displayed. This helps users know that the page is ready to be used and shows what the bottom section of the explorer is for.

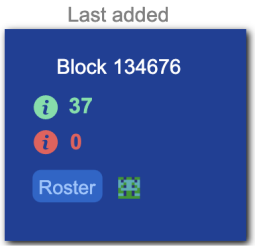

Figure 10: The last added block of the chain with its information

The display of the last added block offers on-hand information such as:

- The block's index along with its hash in a tool-tip.
- The number of validated and rejected transactions of that block.
- The roster's hash illustrated by a blocky and a tool-tip situated on the `Roster` displaying the participating conodes

For this feature, we have created a new class called `LastAddedBlock`. That is where the last added block of the chain is fetched and where we insert all information on the block (Figure 10).

3.2.6 Displaying blocks starting from the last added block of the chain

The previous explorer displayed blocks from index 20 on-wards on the load of the page. Since we now have a dedicated space for the last added block of the chain, we wanted to keep the explorer consistent and start displaying the last validated block going down to the first blocks on the chain. By doing so, we realized that there was an issue in the Byzcoin implementation. In fact, forward links towards to end of the chain are broken. This is due to forks that have been made in the past to resolve other bugs, but links had never been reinstated correctly. Although links have been repaired now, this made the choice of highlighting the last added block of the chain even more relevant, since some blocks at the end of the chain could not be displayed. Having a dedicated space helps visualise the size of the chain and where the bugs are located.

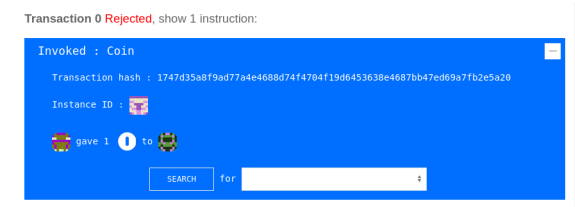

Figure 11: An example of an instruction accordion

3.3 Block related implementations

3.3.1 Improved the layout and modernized the design

After the colour, we changed the layout of the page. We split the bottom part of the screen in two columns. The left one is used for anything related to the characteristics of a block such as the height, the verifiers and the links emerging from that block. The other one is used to display the details of every transaction stored in the block. We kept the "accordions" as they help the users to focus only on what they need. A lot of fields were rewritten as they were weirdly worded or not consistent with the rest.

3.3.2 Reformatting instructions arguments

Instructions were really opaque to a lot of users. It was hard to infer what an instruction was used for. Arguments were also not decoded properly so they were unreadable. This motivated the creation of a beautifier in the ByzCoin client. Columbus now makes use of this beautifier, on top of this some instance types that were supported at the time of the implementation have a specific way (Figure 11) of being displayed to make them even clearer for the user. It was hard to see if an instruction was either accepted or rejected, therefore we added proper highlighting to "warn" the user that something had gone wrong.

3.3.3 More fields

We added some data that was deemed important by the users, such as block validation time, heights of blocks, or the total number of transactions. We made them as little intrusive as possible as the page was already cluttered and these characteristics are not "the selling features" of the explorer although they're nice to have. To do this, we've for example used lighter greys and reduced the weight of the font. This way, the user unconsciously categorizes the field as less important, and their gaze is not attracted to it.

3.3.4 Interactive icons

As mentioned before, apparent hashes were replaced by nicer icons such as badges or images (Figure 12). They provide the same data to the user while taking less space and giving a cleaner look to the interface.

Figure 12: Various examples of interactive icons

3.4 Instance tracking

Byzcoin uses instances to model persistent objects such as Coins or Darcs (authorization handlers). The ability to review all the modifications that have been made to an instance is important to the people working on Byzcoin. The instance tracker is the tool provided by Columbus to suit this need.

3.4.1 Debugging the initial implementation

When we started the project, the groundwork for the feature was already there but it produced incorrect results. Only some Invokes were listed, Spawn and Deletes were simply not appearing. Moreover, some Invokes were missing for weird reasons. The approach planned was to first review the old implementations, refactor it, and then debug it. As it turns out, refactoring turned out to be enough. The problem was that the previous implementations didn't account for the asynchronous nature of the server answers. After some clever refactoring, most of the bugs were ironed out, the remaining ones were very easy to spot and fix.

3.4.2 Reactive design

Previously, the waiting time when searching for instructions related to an instance took place on another view. There was no reason to have it appear on the main view so we fixed it. This allows users to continue browsing the chain while waiting for their query to be answered (Figure 13).

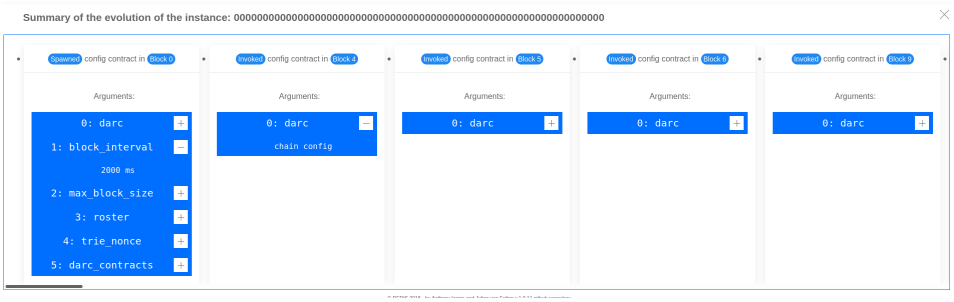

Figure 13: New design of the instance tracker

3.4.3 Partial chain browsing

Browsing the whole chain can be quite long. To allow a more dynamic experience, we worked on the code so that the results that were already downloaded would show up even if the users aborted the query. We went a bit farther and gave the user the option to only fetch a given number of instructions. As sometimes, only the first few instructions related to an instance are useful, not necessarily a thousand others.

3.5 Search bar

The initial explorer limited users to browse around the first displayed blocks of the chain. The only way a user could go from one block to another was by loading all the blocks between the first displayed blocks of the chain and the wanted block. Implementing a search bar enabled users to jump to any block on the chain easily. In this way, a user can get information on any block instantly without having to wait for all blocks to load from the client, which depending on the distance between blocks could take a very long time. When a block is searched for, the chain translates to the given block in order to visualize its position in the chain, and all of its information is displayed. But to keep it fast and simple to use, the initial design of the search bar automatically detects what kind of query the user is trying to do; either an index search, a hash search, or an instance tracking (Figure 14).

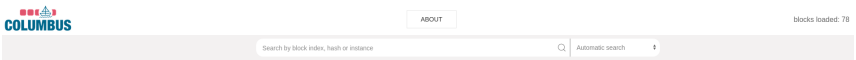

Figure 14: Currently implemented search bar

3.5.1 ID search

Block indexes allow users to visualise where the block is situated on the chain but do not give any further information. Searching by block index helps users

get all the information about a block easily.

3.5.2 Hash search

Block hashes do not give any on hand information about the block nor its position on the chain. By enabling hash searching, users can now visualise the block on the chain and view all of its characteristics directly.

3.5.3 Instance tracking

This feature is already discussed in the precedent section. The option to track an instance in the search-bar was implemented solely for added usability. Indeed, a lot of users had a hard time noticing the button under the transaction details column.

4 Intermediate feedback and adjustments

4.1 Feedback from the lab

As we were finishing a part of the features planned. We went for another round of usability study, this time with the DEDIS Lab. We used a Google Form[1] to streamline the study and be less bothersome for people to participate.

4.1.1 Protocol

Firstly, the user had the choice of selecting their level of knowledge on ByzCoin. Depending on the choice they made, the challenges proposed were slightly different and tailored to test common interactions one would have with the explorer. Some of the challenges consisted of:

- What is the height of block 48? And where does the link emerging from block 48 go?
- Starting from block 49, what is the minimum number of blocks you have to traverse to get to block 70? (endpoints included)
- Consider the instance invoked in the first and only transaction of block 48. What is the id of this instance? How many times is it invoked again between block 100 and block 180?

Once the user completed a challenge, we wanted to know how they proceeded and how long it took them. We then asked the user to rate their interaction with the different features and finally, to give us feedback if they had any.

[1] `https://docs.google.com/forms/d/1JHKTgxyXSPAK5vTHzYTxzLA7crwLvszv4yec-sKx0XU/viewform?edit_requested=true`

4.1.2 Results and feedback

The tested users consisted of mostly intermediate and expert users from the lab. The features we developed were appreciated but sometimes overseen. In general, users managed to complete the challenges correctly but did not do so with the anticipated tools. We realized that we had not created enough interaction between the platform and the user. Overall, users were fast and found their interaction quite pleasant. Another recurring comment we received was about the visibility of features and incomplete features. This prompted us to add more tool-tips, hovering action and pointer interactivity on the explorer. These changes were added with respect to the various principles explained in the section below.

4.2 Principles of design

The intermediate feedback made us realize that in order to fix some of the issues we were facing, we needed to approach our interface with a better grasp on design usability. In order to have a better toolset, we tried to incorporate some famous principles of design. [10, 11, 12] However, the points that were made were often contradicting. Some were, for instance, advocating for as many visual representations as possible while other were encouraging to keep designs as minimal as possible. For each principle we found, there often was another principle in complete opposition to the first one. What we gathered is that principles were only here to help us identify what problem we were facing and how to solve them. We should not blindly follow a set of principles but pinpoint our issues with them and balance them in a way that we deem appropriate.

4.2.1 Cognitive load and efficiency

It is hard to both please expert Byzcoin users and newcomers. The firsts, expect to quickly find all the technical data they need. They already know how Byzcoin works and should not be slowed down by features that are not related to what they are looking for (e.g. a welcome screen with a tutorial). Beginners on the other hand should not be presented with more information than what they can handle or they risk being overwhelmed and unable to find the information they are looking for.
The metric we used to measure the quantity of information that was presented to the user is the cognitive load. This concept theorized by John Sweller [8] and expanded by many other experts in usability is used to measure the number of pieces of information a user can "store" in his working memory at a given time (like a processor's L1 cache). This threshold is estimated to 5 for

beginner users and up to 9 for expert users. As such, users should not need to think about more than 5 pieces of information in order to interact with Columbus. Neither should the site overload the user with too much information.
As we were adding more and more features, we had to be careful when choosing where we would display them so they were not one more distraction on the main panel. We also kept using "accordions" as they allow the users themselves to tailor how many different fields of data were displayed.
There is however an obvious pitfall with too many sub-menus. Features that are "too deep" in the sub-menus hierarchy are either hidden and never discovered by users or require the user to actively think where to find the feature (thus increasing the cognitive load). One example that surfaced from the intermediate feedback was the instance tracking button that took a long time to notice. In our attempts at reducing the clutter of the space, we had to be careful to keep the information and the features easy and fast to access.
There was a balance to be found between having an interface that only allows a few actions at a time and an interface that displays a lot of information at the same time.

4.2.2 Affordance and simplicity of design

We added a lot of interactions to the Columbus page, such as the clickable badges, blockies, the search bar, and so on. Once again, they went a bit unnoticed. The problem was not that the blockies or the search bar went unnoticed, they were presented in uncluttered space and had bolder colors so they would draw the user's attention. The issue was that once users noticed a blocky, nothing was allowing the user to understand that there was an interaction with it. The components of our interface were not communicating their functionalities to the user. This principle is called affordance [9]. Doors, for example, often indicate if they are to be pushed or pulled simply by removing handles in the first case. They are several ways we thought of in order to increase the affordance of our design. For example to indicate that a user could click on a blocky, We could have added more tool-tips, icons next to it, more highlighting, and so on. The downsides of all those solutions are that they add a lot of visual noise and complexity to our design. If the designs become too complicated the user will not want to engage with it as he will assume that if a feature needs a lot of explanation, it's going to be a struggle to use.

4.2.3 Consistency and Identifiability

In our attempt at removing hashes from the page, it was important to us to properly communicate to the users what purpose each field served. For exam-

ple, we used blockies to represent data that is "unique" such as a hash or an ID, because the "faces" provided by blockies are unique and identifiable. But by fixing a problem, we ended up with a new one. Instance IDs and users now were both represented by a blocky, which meant that some users would have a hard time conceptually distinguish two very different fields. Users expect all blockies to behave consistently, they should all serve the same purpose and offer the same interactions. But if we were to add yet another type of icons to our interface, we feared it would get in the way of identifiability and hinder the learning curve by providing too many different types of fields. Another insightful example was found with the instance tracker. As it was rendered as a timeline just like the Skipchain visualization at the top of the screen, users expected the instance tracker to behave like the chain. They wanted to be able to click and drag to navigate the results of the query and were frustrated because they couldn't.

4.3 Ensuing changes

- **Pointer interactivity**
  In most cases, users did not realize they could click on many of the interactive fields. It meant that the icons were not properly communicating their functions by their designs. To fix this, we've added some affordance to our designs with highlighting and changing mouse icons (Figure 15) to indicate to the user what he can interact with. This includes blockies, arrows, links, blocks, the chain, and the instance tracker.

Figure 15: Different mouse icons used to notify the user of a possible interaction

- **Versatile search bar**
  The users were confused as to what the search-bar was capable off. It's very minimalist design probably convinced the users it wasn't a very powerful tool. Like a lot of other features, it was lacking affordance. Too emphasize it a bit more and indicate to users that the toolbars were enabling multiple types of searches, we added a drop-down menu next to it. To validate this change, we have done A/B testing in our final UX studies to test which implementation was more efficient. The A group was provided with the "automatic match" search bar, and the B group was provided with the search bar with the drop-down menu. Both groups were made of 2 beginner users, 2 intermediate users and 1 expert user.

- **Translations on the chain**
  Users didn't automatically notice that the search bar was working, because only a flash message was showing. If the selected block was not

shown, it could be quite hard to see that it was indeed selected (although it was possible thanks to the block index field). To give the user more feedback, we made the chain view translate to a block when it was selected through any other means than directly clicking on it. To make it even more obvious, we've animated the translation.

- **Improved instance tracking navigation**
  Although they are a bit different in design and in the feature they offer, a lot of users expected the instance tracker to behave in the same way as the top chain. To make it more consistent, we've added the same click and drag feature to navigate, along with a changing mouse icon.

5 Conclusion

5.1 UX Validation

In order to evaluate our work, we organized another round of UX studies. This time, the goal was not only to get direct feedback on our features, but to see how the problems we faced at the beginning of the project evolved. For this reason, half of the participants were already familiar with Columbus whereas the other half were not introduced yet. The protocol used was as follow :

- 3-4 minutes of free roam without any indication or direction. This helped us see which features the users were naturally engaging with and the ones that didn't attract them as much. It also allowed us to see which concepts of the Byzcoin architecture could be inferred solely from the interface.
- 2 minute of a brief explanation of the functioning of Byzcoin, but no guidance on Columbus itself
- The tasks used in the intermediate feedback form were used again, the users had to solve them without any help. Thanks to this step, we could study how the users could relate the explanation they had of Byzcoin to their experience navigating with Columbus.
- 3-4 minutes of feedback supported by questions such as the one asked in the first round of UX studies. This helped us determine how the problems Columbus initially faced when we took over evolved.

Generally, users were satisfied by the aesthetics of the explorer. They found the explorer pretty and its information spread equally. Although the sliding of the last added block helped the user know the page was loaded and ready to use, the fact of being able to zoom and slide the chain takes a few extra seconds. The validation study was made up of 13 people, 6 beginner users,

7 intermediate users and 2 expert users. We noticed that the beginner users tended to first analyze the graphical elements of the UI and try to deduce from there what the other fields meant. Intermediate users however dove first in the two columns and tried to find what they implied on the graphical representation of the Skipchain.

5.1.1 A/B Validation on the search bar

Starting from the upper half of the explorer, the search bar was tested in two different configurations.

Figure 16: The drop down search bar

Figure 17: The "automatic" search bar

The two search bars both had both success and letdowns. The automatic search bar was, as stated before, often overlooked. But once a user understood its functionalities, it became in some case their most used tool. It is fast and efficient. It was mostly adopted by beginner users that didn't really understand all the Byzcoin vocabulary, as it allows them to perform searches without having to completely understand what they were doing. Expert users really liked it as well because they found it more convenient. So much so, that most of their navigation was based on the search bar. The drop-down search bar, on the other hand, was most appreciated by the intermediate users. The drop-down menu allows users to clearly see what it is capable of. But users reported it was cumbersome to use because they had to think about the right mode before using it.
Instead of having to choose between one of the two search bars, we decided to combine them. We kept the presentation of the drop-down search bar because the users were more drawn to it and it indicated it's purpose more clearly. But we've added a "search for anything" mode which is selected by default to keep the efficiency of the automatic search bar. Initially, it takes more clicks to understand how the search bar works but it rapidly pays off. Newer users are more guided, but it is not penalizing the efficiency of the power users. Users sometimes thought the search bar was not working when searching for a block hash or an index. Indeed very little visual feedback was given to them except the flash message and a change of some fields in the columns. We fixed it by animating a translation in the chain whenever a new block is selected through the search bar or one of the links.

5.1.2 Chain representation

Concerning the chain section, users that had seen the explorer before any modifications really enjoyed the display of height and arrows as they now know what they represent. However, users (mostly novice) that were testing the explorer for the first time did not immediately understand why blocks had different heights. They speculated that they were perhaps "less important" or "smaller". Looking for that information is fastidious as one would have to compare the block information and notice that the "height" number changes. The more advanced users however picked up the correlation between height and link number intuitively. Along the same lines, beginner users didn't really understand the purpose of the forward and backward links. However, they were able to navigate the chain by using them. Some intermediate users also had issues conceptualizing the links. Some, for example, thought that it was a representation of the forks in the chain. Expert users however thought that it was a nice representation of the actual Skipchain. Users seemed to discover the features of the chain faster than on the first iteration of our UX studies. Almost all users found out they could click and drag the chain and immediately clicked on a block. Many users also noticed that the links between blocks were clickable as well but they complained that they were required too much mouse precision to click on them as they are very thin. We noticed that the users that based the most their navigation on the chain representation tended to be novice with blockchains and explorers. Whereas advanced users almost completely discarded it.

5.1.3 Block and transaction details

The page is only displaying the Skipchain when starting. It is the user actions that gradually build the interface. This is nice as it allows the users to build a mental model while they progress in the interface. However, the newer users had troubles understanding the conceptual difference between the two columns by themselves, and did not know in which column they should look for a specific piece of information. However, the users that were already familiar with the interface reported that they felt that it was more efficient and insightful than before. Accordions were appreciated, newer users could sometimes understand the hierarchy between transactions, instructions and arguments thanks to it (though it often was not enough). Intermediate users liked that they could easily hide information that was not related to their task. Most users felt that the information was not cluttered and said that once they understood the whole layout of the page, they found the information they were looking for without a struggle).

5.1.4 Blockies and interactive fields

Blockies were widely welcomed by all users. Even the less experienced users understood that they were associated with a hash and that they were interactive. Some users were confused that some blockies were rounder than others, meanwhile few realized that rounder blockies were related to a user and square blockies are related to an object in the chain. Expert user and the most advanced intermediate users really liked the blockies as it allowed them to easily compare some instances and recognize some important instances (e.g the the config instance) or active users. Thanks to the pointer changes, almost all users figured they could click an icon to copy hashes to their clipboard.

5.1.5 Instance tracker

The instance tracker was still less noticed than the other features, but it still was a wide improvement over the other studies we did. Users navigated through the query results very intuitively, they applied what they learned with the Skipchain representation to great results. Some users wanted this feature to be expanded further, and wondered if there was a way for them to find more information about a given instance.

5.1.6 Feedback synthesis

Overhaul, visibility of the features seem to have been improved. Most users were able to find the tools they needed rapidly and would then adopt a workflow that they enjoyed (for example, expert users relying mostly on the search bar, novice users navigating with the visual representation and intermediate users using a mix of everything).
Unfortunately, some features seem to still be overseen for two main reasons:
First, they are simply not visible. Particularly, users did not realise that the URL reflected the state of the chain. Therefore, they did not know it was possible to start loading the chain from any point and that they could use the backward and forward arrows of their browser to get back to previous blocks visualised.
Secondly, users do not tend to use them as they do not know what they do nor understand what they are for. As an example, the instance searching feature is a very powerful tool that is not used by novice or intermediate users. The reason being is that they simply do not understand what it is for.

There is still room for improvement! We enumerate different ways of improving these problems in the section below along with some other features that we feel could make Columbus a greater tool.

5.2 Potential future for Columbus

We globally managed to implement all initially planned features, although it seemed very ambitious at the beginning. However, there is still room for betterment and additional features. The first general elements we could improve are the points mentioned by the second round of feedback we received from users and the DEDIS lab. Due to the short time span of the project, some secondary features we had planned to implement did not make it on the explorer, this could perhaps inspire future students working on Columbus. Following are a few improvements that could be made on top of the existing features:

- Adding information to blocks on the chain, in particular when they are zoomed in to the maximum. In this manner, the user would have access to on-hand information about multiple blocks at the same time. This could also serve as an illustration for users that do not know what a Skipchain is. For example, displaying the heights of each block could explain why blocks are of different heights on the chain.
- Modifying the last added block space by the current selected block. At the current time, having a dedicated space to highlight the last added block of the chain makes sense since we are unable to load blocks further than indexes 118750 due to an error in the Byzcoin implementation. However, when this issue will be resolved, displaying the currently selected block in the dedicated space may offer more on-hand information directly without having to search through the block information.
- Customizing the beautifier even more once its implementation is done akin to what have been done with coins
- Adding more search mode for the search-bar (User ID for example)
- Computing statistics about the blockchains (most used transaction, average validation time, etc...)
- Designing a better display of the roster and allow the use of a different roster
- Add more features to the instance tracker, such as a way to consult the most recent state of an instance easily.
- Implement a "user tracker" which displays all the transactions that have been issued by a user. It could also show how many coins the user possesses, or his Calypso access rights.
- We focused our effort to build an explorer that is versatile and comprehensible for all categories of users. Indeed, perhaps the addition of a tutorial

to give some context to ByzCoin and Skipchains for inexperienced users could be helpful.

5.3 Improvements of the Byzcoin implementation

Implementing new features to the explorer allowed us to discover some bugs that were unknown in the Byzcoin implementation. Columbus is not only about visualising Byzcoin and its underlying skipchain, it is also a tool to help strengthen the Byzcoin implementation. During the implementation of the chain, we wanted to display the chain from the last added blocks instead of from block 0. This is how we discovered that forward links towards the end of the chain were broken. We weren't able to fetch from the client blocks from that point onward. This is due to forks that have been made in the past to resolve other bugs in Byzcoin and links have not been repaired after that. The following are repairs that have been made to the client:

- Add bcadmin db optimize command #2424[2]
- Adding SkipchainDB.GetProofFromIndex #2423[3]
- Optimize chain optimization #2422[4]

Some QoL improvements have also been implemented to the client such as a beautifier. Instead of delegating the decoding and parsing of instruction arguments to the user of the Cothority API, the new beautifier decorator takes care of it. #2401[5]

5.4 Personal retrospectives

5.4.1 Sophia

We started the project by defining the features and the various steps to attain the end-goal for Columbus. It was actually quite thrilling to visually realise what had been planned and to see the explorer take shape throughout the semester.
During the first weeks of implementing the features, it was quite difficult to engage and adapt to the already existing code written by the students working on the project before us. The challenges that were given to us before-hand helped with this matter however, a lot of work was still to be done to get familiar with the entirety of the stack. I had only done very few applications

[2]https://github.com/dedis/cothority/pull/2424
[3]https://github.com/dedis/cothority/pull/2423
[4]https://github.com/dedis/cothority/pull/2422
[5]https://github.com/dedis/cothority/pull/2401

of front-end in the past and despite trying to learn as much as possible before the start of the project, it was still difficult to juggle between learning how to use new frameworks and tackling the implementation of the various features. Progressively, I became familiar with the stack and could move my focus onto more advanced aspects of the code rather than the basics. In this way, I managed to implement all the planned features, even though it seemed very ambitious at the start.
Furthermore, another challenging point was working on a project that had an external component to it. Particularly, we had to engage with the Cothority client, which had some implementation issues. I indeed spent a lot of time trying to fix errors encountered while implementing features, relying on documentation that was perhaps not sufficient for my understanding at the start. This made identifying where bugs came from more complicated and time-consuming.
However, Columbus is the first development project of this span I ever implemented, I am aware that these are important elements that are dealt with regularly. In this sense, I feel that encountering these problems was really enriching for my future work as an engineer.
Additionally to what has been stated above, I appreciated having a social aspect of the project. Interacting with the different sorts of users for User Experience feedback or with actual potential users of the explorer such as the DEDIS lab was very insightful.
I also learned how to work as a team along with Lucas. We had to find common ground on features to implement and how we wanted to spread the work between ourselves. We also had to merge our individual work several times and collaborate with tools such as Github in order to align our different features.
Columbus was an extremely enriching project to realise, particularly because it has helped bring improvements to the Byzcoin implementation and it has the potential to actually be helpful to engineers (and others).
Despite the various challenges I had to face, working with the DEDIS lab on this project wasn't only an enriching experience but also enforced my interest in decentralised technologies. Moreover, Noémien's supervision was extremely valuable, he gave me a lot of freedom to advance on the project and at the same time was always available to help.

5.4.2 Lucas

Prior to Columbus, I had no consequent experiences with neither front-end development nor blockchains. Which made me a bit skeptical at first when approaching the project. Would we be able to produce a great contribution simply with front-end developing, especially on a web app that already ex-

isted?
Diving into the code headfirst would have likely produced a bad result. Therefore the first weeks, I read as much as I could about Byzcoin and basic rules of design so I'd get a better grasp. Because everything I read about design was always focusing on the users, it seemed clear to me that we were going to have to integrate the users in our development process as much as possible. Our first round of UX studies, was definitely sub-optimal, but it turned out informative enough. Most importantly, I realized that the next ones would need to be gingerly organized and thought thoroughly.
Once our first assessment was made, designing the first sketches and ideating features were a lot of fun. I also enjoyed the challenge of quickly learning a dev stack and taking over a code base, it was a great opportunity to learn what I believe is a critical skill in the line of work I'm aiming for. One of the strong aspect of front-end development is that we could see the effect of our work on the page very quickly. Things such as simply changing the colours of the interface could produce great results with little effort, having direct visual feedback was really gratifying.
What was even more gratifying than changing colours, and one of the best memory I have with the project, is having one of the engineers who developed Byzcoin endorse the project. It pushed me harder to strive for a great product and motivated me to go even further with the elements of design.
Learning about more advanced principles of design to get the finishing touches right felt sometimes a bit underwhelming on the implementation side (*back to changing colours...*). But it proved to be essential when we did our validation studies. Seeing how very small improvements could allow us to efficiently guide the users proved rewarding in the end.
Having Sophia as a teammate was very valuable, our cooperation felt very efficient, we were able to make decisions fast and thanks to the combined effort of our hard work, we were able to push Columbus way further than what I anticipated. Noémien had a critical role in our work as well, whenever we needed him, he provided with fixes and improvements to the cothority API which saved us time. He gave us a lot of freedom in the features we wanted to implement as well as on our workflow, which I really appreciated.
Overhaul, Columbus was one of the biggest projects I've worked on, and one I'm the most proud of as well. I can't wait to see how Columbus is going to evolve next semester and I hope it'll be a valuable tool for the lab !

References

[1] DEDIS Lab, *ByzCoin Repository*, `https://github.com/dedis/cothority/tree/master/byzcoin`.

[2] DEDIS Lab, *SkipChain Repository*, https://github.com/dedis/cothority/tree/master/skipchain.

[3] DEDIS Lab, *Cothority repository*, https://github.com/dedis/cothority

[4] Bryan Ford *How Do You Know It's On the Blockchain? With a SkipChain.* https://bford.info/2017/08/01/skipchain/ 2017.

[5] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, Bryan Ford, *OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding*. https://eprint.iacr.org/2017/406.pdf, 2018

[6] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, Bryan Ford *Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing*, 2016

[7] Noémien Kocher, Anthony Iozzia, Julien von Felten, Sophia Artioli, Lucas Trognon. *Columbus production build.* https://wookiee.ch/columbus/, 2020.

[8] John Sweller, *Cognitive Load During Problem Solving: Effects on Learning*, 1988

[9] J. J. Gibson . *Reasons for Realism: Selected Essays of James J. Gibson. Resources for Ecological Psychology. L. Erlbaum*, New Jersey, 1982

[10] Ben Shneiderman, *Designing the User Interface*, 1986.

[11] Jakob Nielsen, *Heuristic evaluation*, 1994.

[12] Don Norman, *The Design of Everyday Things*, 2013.

[13] Miguel Castro and Barbara Liskov, *Practical Byzantine Fault Tolerance*, 1999.

[14] *Columbus III ToDo*, https://github.com/dedis/columbus-united/projects/1

[15] *Columbus III repository*, https://github.com/dedis/columbus-united/