Basic conditions for a mathematical forced decentralized network protocol (MfDN-Protocol)

https://blog.carolin-zoebelein.de/2020/06/mfdn-protocol-basicconditions.html
Sun 07 Jun 2020 in Decentralization, Carolin Zöbelein

The following was published as research notes on https://research.carolin-zoebelein.de/Public/Publications/Notes-Basic-conditions-for-a-mathematical-forced-decentralized-network-protocol-MfDN-Protocol.html at 2020/02/20, for the first time.

Introduction

In this notes, we will discuss the main basic conditions which are necessary for a mathematical forced decentralized network protocol, short *MfDN-Protocol*.

What does MfDN-Protocol mean?

Decentralization is given by distributing computation power, or by 'sharing splited' data or information with each other on a technical, computer science level. All of this could theoretically be happen also on only one device. Also the common shared data, can be centralized without any problem on only a few or even one machine.

Hence, *Mathematical forced dececentralization* means a protocol which follows a mathematical structure which needs mandatory decentralization to be able to work. To understand this concept better, we want to introduce it by having a look at the differences between a centralized service, a classical decentralized service and a mathematical forced decentralized service, at first.

Centralized service

Situation. The service is maintained by a single party on a centralized computer network.

Main design points.

1. One single party maintains the service
2. The service runs on a centralized computer network
3. 3 Every user of the service have to connect to this centralized network
4. 4 Data is saved centralized
5. 5 Traffic is centralized to the service computer network

Disadvantages. The service itself, the collected data and the generated traffic, the user traffic as well as the service traffic, is centralized and hence in the power of the single party who runs the service. This makes acquisition of data and censorship of information very easy. Additionally, centralized networks have limited scalability.

Classical decentralized service

Situation. The service is maintained by a bunch of parties on a decentralized computer network.

Main design points.

1. Several parties maintain the service
2. The service runs in a decentralized computer network
3. Every user of the service connects to the decentralized network
4. Data can be saved decentralized
5. Traffic is centralized to the members of the whole decentralized computer network

Disadvantages. Even if we have a decentralized service, a centralization on a deeper level is still possible. Services itself, can be designed as centralized services itself, only running as several instances on several computers, data can, but not have, to be saved decentralized and even if users connect to a decentralized service, their individuell traffic can still be distributed within the network in the same way like within a centralized network, without mandatory anonymization and identifiable information about the origin. Their individual traffic can be still centralized, too.

Mathematical forced decentralized service

Situation. A mathematical forced decentralized network protocol is maintained by a bunch of parties on a decentralized computer network. The service is run on the top of this protocol.

Main design points.

1. Several parties maintain the protocol network for the service
2. The service runs on the top of the mathematical forced decentralized network protocol in a decentralized computer network
3. Every user of the service connect to this protocol which coordinates the decentralization itself
4. All data is decentralized saved, organized by the protocol
5. The protocol manages the partition of user and server traffic within the network
6. The protocol is designed in such a way, that each service which runs on the top of it, can't be centralized, without loosing the ability to work. This forced decentralization is given by mathematical design aspects of the protocol, not by the technical, physical decentralization on computer level, alone.

Disadvantages. A given service has to be designed in such a way, that it is able to work on top of our given protocol (the protocol should support as much service designs as possible but it will be necessary that a service fulfills at least a minimum amount of design constraints.

Critical design questions

For our protocol, we have to take care of the following critical design questions:

1. Defining the minimum amount of design contraints which a service has to fulfill
2. How to take care of avoiding a 'fake' decentralization by e.g. running several virtual machines on the same centralized server?
3. Best way of handling and coordinating network members, synchronization, concurrency control, ...
4. Best way of handling recovery problems because of user errors, application errors or partial system failures
5. The given hardware and infrastrucutre have to be able to handle the amount of traffic, synchronization, ...

Main decentralization points

If we look at the mentioned concepts of networks, it crystallizes the following main decentralization points for our MfDN-Protocol:

1. A mathematical forced decentralization of the running service
2. A mathematical forced decentralization of data
3. A mathematical forced decentralization of user and service traffic

We will more clarify this in the further research discussion notes. For the moment, we will simple keep this in mind, at first.

Conclusion

After having a look at the basic conditions for our mathematicial forced decentralized protocol, we want to put this in a more clear design in the next notes. There we will discuss the basic protocol design and the deeper parts with the mathematical aspects of our work.

Acknowledgement

Special thanks to the individual donators for supporting this work.