CS6675 Project Proposal

Students:

Akshat Shetty, Adhrit Shetty, Jiaxiang Zhu, Jin Hyung Park

Instructor:

Prof. Ling Liu

(The Initial Proposal / Plan for the Project)

MinCast: A New Strategy to Improve Broadcast Network

1. Motivation.

Blockchain as a technology has gained prominence over the world especially in the form of cryptocurrencies such as Bitcoin and Ethereum. While cryptocurrencies are currently the most popular application of blockchain, a lot of research effort is being put into utilizing the distributed ledger for a number of other use cases such as supply chain management and asset management. A big part of the blockchain network is the broadcast of blocks as part of a distributed consensus protocol. Since blockchain networks operate as unstructured peer-to-peer (p2p) networks**[1]**, most of the communication occurs through broadcast. On paper, this seems to be a working solution. However, the nature of its unstructured network leads to a lot of duplicate messages being introduced in the network, resulting in high messaging overhead. In order to resolve this issue, many networks limit their broadcasts to only a subset of neighbors. This is not a good solution as it increases the block propagation delay that may have an impact on the performance of the system. Therefore, optimizing the block propagation procedure has been identified as critical to the scalability of blockchain. These optimizations will help achieve higher transaction rates and reduce the possibility of fraud in the system.

Our group is highly interested in this topic, and have read through a few research papers about improvements in this area. For instance, one of the most recent and important researches, named "*Kadcast***[2]**", introduces a new peer-to-peer network protocol designed specifically to resolve the block propagation issues. It is based on a distributed hash table called "*Kademlia***[3]**", and achieves higher performance, reliability and security than traditional broadcast designs. It significantly improves the block distribution with less overhead introduced in the process.

However, despite all the benefits in new protocols like Kadcast, there are still many issues with broadcast networks, with some of these research introducing new problems in one way or another. For example, Kadcast itself uses UDP as its transportation layer protocol, which focuses more on speed than data integrity. It is possible to receive corrupted data in the process of data transfering. Because of this, our group decided to do our own research, and bring up a new strategy / protocol called "*MinCast*" to resolve the bottleneck of the broadcast network. It is aiming for minimum broadcast on demand, as well as quick transfers between nodes within a peer-to-peer network. This would serve as both a challenging task and a learning experience for us to build a broadcast network from ground up.

2. Related Works.

Numerous strategies have been proposed to improve the scalability of blockchain. This project is confined only to block propagation strategies. For example, Erlay**[4]** proposes alterations to Bitcoin's transaction relay protocol which help decrease bandwidth consumption. However the decrease in bandwidth is accompanied by an increase in propagation latency.

Other strategies include reducing the amount of data which needs to be propagated. For example Velocity**[5]** makes use of Fountain code, a kind of erasure code to do the same. Basically instead of a single peer sending the entire block data, multiple peers send partial fragments of the block. This helps reduce the load on that peer and also allows a receiver to request a single block from multiple peers eliminating a single point of failure. It argues that simply reducing the size of the block cannot help us achieve high scalability.

Some solutions like bloXroute**[6]** involve the use of third-party relay networks. It employs a Blockchain Distribution Network (BDN) which is basically an overlay network of globally placed servers that improve the rate of block transfer between peers. It makes use of advanced network optimization techniques and caching to speed up the rate of block transfer. It further has mechanisms to ensure that the BDN is neutral to all peers in the network. It is not very useful to our research since we cannot go about deploying global servers to improve block propagation. However, it does provide good insight about neutrality maintenance and improvements on the broadcast process.

Based on our research we found most promise in a block propagation strategy called Kadcast**[2]**. It makes use of Kademlia, a structured peer-to-peer overlay in order to achieve tunable message overhead and latency. On average, Kadcast distributes blocks 30% faster than deployed vanilla blockchain protocols. For this project, we will also consider other broadcast strategies outside of the blockchain (i.e. El-Ansary et al**[7]**) as they also utilize some forms of structure peer-to-peer network.

3. Proposed Work.

The initial stage of the project involves using network simulators such as NS-3**[8]** to simulate a blockchain broadcast. Our improved strategy (MinCast) will be deployed with NS rules to make comparisons with vanilla broadcast (and with other existing strategies later). This is a proof of concept work before the actual implementation of our strategy in real servers. We plan to spin up 6-10 nodes in the simulator to represent the clients within the peer-to-peer network. Since we only focus on the network side of things, we might not need an actual blockchain on top of the network. Still, we will deploy open source offerings on our nodes if needed. We plan to utilize the ideas of structured network in replacement of the standard broadcast flooding methods, so that our network operations can be more efficient, more reliable, and simpler with lower redundancy / overhead.

Once we proved that MinCast rules are working in a network simulator, we will then port our code to actual servers for real-life scenarios. Since Kadcast is the latest solution to broadcast networks in the field, we initially planned to utilize / tweak their source code in Dusk Network**[9]** for MinCast. However, due to the complexity nature of the existing code base, as well as the fact that it was implemented in Go, we decided to change our direction. We plan to use Shell / Python / JavaScript to implement MinCast in actual servers, with improvements over the vanila broadcast network and others. In general, we want to tweak / modify our network so that the broadcast can be carefully adjusted to scale, and controlled as if we are running on a centralized network.

Since we are still in the early stage of our project, steps of the implementation might get changed in the process. We will mention some details in the evaluation / foreseen risks and plans section of this proposal.

Implementation Details:

- System:

  (1). Proof of Concept Stage:

  We will use NS-3 to spin up multiple nodes and simulate a network for broadcast. We will quickly adjust the rules and settings to simulate our approach of MinCast. If we are able to show any improvements in our implementation compared to vanilla broadcast networks, we will collect the data and enter the second stage of real-life server testing (more evaluations and performance comparisons in section 6).

  (2). Real-Life Scenarios Stage:

  We will deploy the new MinCast network code in one of the following approaches:

  - We build a docker host on our local home network, with 30-60 small containers running in parallel. Each container will be assigned a maximum of 1.0 CPU share, 128MiB of soft memory limit, and corresponding storage space of Solid State Drive. We may choose among Alpine / CentOS / Ubuntu for the operating systems. There are many benefits with this approach, including free of charge, easy to control, and reproducible environment. The issues being that network IP addresses are private, requires a decent base host machine, and limited network tweak and manipulation.

  - We use Amazon Web Service to provision VMs. All of the VMs will have a t2.nano instance type. Each of them has a maximum of 1vCPU, 500MiB of memory, and an EBS attached to it. We will choose among Amazon Linux / RHEL / Ubuntu for the operating system. The benefits of this approach are easily managed public networks and IPs, full linux kernel support, and world-wide availability. The shortcomings including AWS fees for management, harder to manage compared to docker-compose, and heavyweight.

  Since we are not planning to do any visualizations, we have decided not to include a GPU in our setup. All of our data and chart will be manually populated through excel sheets.

- Possible Open-Source Blockchains: We are currently looking at two open-source blockchain implementations. They are Corda**[10]** and the CodaProtocol**[11]**. Both of these offerings have docker container repositories on official docker hub, so we can easily provision a stack of nodes running blockchain networks. We might add more offerings or even create our own simple blockchain for testing purposes in the process.

- Tools & Languages: We will use NS-3 and its rulesets for the network simulation in the first stage, and use either Docker or AWS in the second stage. We plan to use Shell / Python / JavaScript as they are known by every member of our team. Plus, we use GitHub as our code repo since it is easier to make changes on the fly and integrate our code there. Several libraries will be used in the process and we will list our comments and findings in the code comments.

- Others: We currently do not need a centralized database due to the nature of peer-to-peer networks. Therefore, we will add more information in the final report.

4. Plan of Actions / Timelines / Milestones.

	Jan. 27	Feb. 3	Feb. 10	Feb. 17	Feb. 24	Mar. 2	Mar. 9	Mar. 16	Mar . 23	Mar. 30	Apr. 6
Team discussing topics / system / implementation / detail works. Proposal prep, setup test bench, and finalize the pre-work for the project.	🟦	🟦									
Prepare meeting with professor, test broadcasts.		🟥	🟥	🟥							
Implementing broadcast ideas, research other shortcuts.				🟩	🟩	🟩					
Test broadcast on the entire network stack.					🟪	🟪	🟪				

Compare the new broadcast with flooding and collect data.						🟦	🟦	🟦			
If time permits, add more data analysis based on different factors.							🟦	🟦			
Write a final report, also prepare for workshop demo with professor.								🟪	🟪	🟪	🟪
Prepare the project demo, and the final representation.										🟩	🟩
Process Documentation.	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

5. Deliverables.

(a). Final Report about building the application, setting up the tests, and validating the results (data tables, graphs, analysis and conclusions etc.).

(b). Support/update document about steps of setting up the demo and its working environment in order to get replicable results. We will also include the changes we made in the process of the project.

(c). Source code (if available as open-source) and documentations of all the tools and libraries we used in the project.

6. Evaluation, Foreseen Risks, and Possible Future Works.

(a). Evaluation:

As previously mentioned, numerous attempts to optimize broadcast networks have been made. However, each strategy came with its own downside. Therefore, we decided to implement our own approach “MinCast” as an attempt to not only resolve the issues that plague the vanilla broadcast (flooding) network, but also improve certain shortcomings that may present in other strategies (see section 2 for a list of related works).

Here are the steps for evaluations:

- Implement MinCast in NS-3 alongside vanilla broadcast network, capture data with simulated traffic, and compare the performance results between them.

- Implement MinCast with language of our choice in actual server stacks, capture data with real-life traffic between the nodes, and compare the data / tables with vanilla broadcast.

- If MinCast is able to consistently perform better than vanilla broadcast, we will then compare our data to results from other strategies (Kadcast, Erlay, Velocity, etc.).

- We will also integrate all the data to determine approximately how much of performance improvement we have accomplished and how much improvement is needed for us to approve them as an acceptable outcome.

- If time permits, we will continuously make improvements and tweaks to MinCast, and perform more evaluations later.

(b). Foreseen Risks:

- Broadcast network has had issues with its performance for a long time, and it is clear that several researchers have already attempted to find a solution. We have great ambition and confidence to bring something new to the table, but may also face a hard wall that our predecessors tried to break through for years.

- In the best-case scenario, MinCast should outperform the vanilla broadcast formula, as well as a subset of existing strategies. However, we may have a hard time showing better results when competing with proven strategies such as Kadcast due to our lack of experience and time. At the very least, MinCast should definitely offer improvements over the vanilla broadcast formula.

- It is possible that our results may not show significant differences in performance due to the scale of our network, either in simulation or in real-life. The maximum capacity for our server pool is around 30-60 nodes, which may become the bottleneck of our testing scenario. With the small amount of servers in our hands, the issues such as overhead and propagation delay might not be as bad as we projected. Thus, the benefits of MinCast may not be direct and clear in our evaluation.

(c). Future Works:

- Possible expansion on the scale of our test pool, up to 100-200 nodes with real-time monitoring tools and reports.

- Create our own blockchain on top of MinCast, and test the performance of the entire network stack, with client-side application released to general public.

- Share MinCast online as an open-source project, and invite public contributors to help maintain and improve the code base.

7. Reference.

[1]. Peer-to-Peer Network, Wikipedia
https://en.wikipedia.org/wiki/Peer-to-peer

[2]. Kadcast: A Structured Approach to Broadcast in Blockchain Networks
https://dl.acm.org/doi/pdf/10.1145/3318041.3355469?download=true

[3]. Kademlia: A Distributed Hash Tables for P2P Networks - Wikipedia
https://en.wikipedia.org/wiki/Kademlia

[4]. G. Naumenko, G. Maxwell, P. Wuille, S. Fedorova, and I. Beschast-nikh, "Bandwidth-efficient transaction relay for bitcoin," arXiv preprintarXiv:1905.10518, 2019.

[5]. N. Chawla, H. W. Behrens, D. Tapp, D. Boscovic, and K. S. Candan, "Ve-locity: Scalability improvements in block propagation through rateless erasure coding," in 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2019, pp. 447–454.

[6]. U. Klarman, S. Basu, A. Kuzmanovic, and E. G. Sirer, "bloxroute: Ascalable trustless blockchain distribution network whitepaper

[7]. Sameh El-Ansary, Luc Onana Alima, Per Brand, and Seif Haridi. 2003. E- cient Broadcast in Structured P2P Networks. In IPTPS '03: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (2003). 304–314.

[8]. NS-3 Network Simulator
https://www.nsnam.org/

[9]. Dusk Network Website
https://dusk.network/news/dusk-network-development-update-november

[10]. Corda: An Open-source Blockchain Project.
https://github.com/corda/corda

[11]. CodaProtocol: A Lightweight Open-source Cryptocurrency.
https://github.com/CodaProtocol/coda