A Decentralized Service Control framework for Decentralized Applications in Cloud Environments

Bram Hoogenkamp, Siamak Farshidi, Ruyue Xin, Zeshun Shi, Peng Chen, and Zhiming Zhao

[1] University of Amsterdam, Multiscale Networked Systems,The Netherlands
[2] {s.farshidi, r.xin, z.shi2, p.chen, z.zhao}@uva.nl

Abstract. Effectively managing decentralized applications in cloud environments using a decentralized control paradigm is essential, as current cloud providers usually only offer a control interface for monitoring cloud infrastructures. This study proposes a decentralized service control framework for implementing the control across various organizations and coordinating collaboration among operators in a decentralized application. The proposed framework allows a consortium of organizations to control a shared distributed cloud infrastructure decentralized reliably. A consensus mechanism within the framework enables mutual coordination between the operators. This mechanism also uses an incentive protocol to enforce pro-active behavior and collaboration. We implement the framework with Hyperledger Fabric, and our experiments demonstrate its usability, reliability, and acceptable performance.

Keywords: decentralized control · decentralized applications · consensus mechanism · collaboration protocol.

1 Introduction

Recently, decentralized applications (dApps) have been employed in various industrial sectors, such as car sharing [12], data management [2], and finance [17]. The enabling technologies for dApps, such as Blockchain-as-a-Service (BaaS), have been included in numerous public cloud providers as part of their service portfolio, e.g., in Azure and AWS [3]. BaaS provides dApps with elastic distributed cloud infrastructures and often is operated using consortium blockchains. Current cloud providers usually offer a control interface for monitoring the cloud infrastructure, which is only for individual cloud service levels but not sufficient for dApps deployed by a consortium of organizations across different providers. Moreover, for dApps in cloud environments, it is only possible to track a failing part of the system, but organizations have no collaborative way to control the different cloud infrastructures. It is essential to implement the control across these various organizations and collaborate to achieve global control. Therefore, a collaborative, decentralized control system on cloud infrastructures

for a dApp is needed. In this research, we mainly focus on the question: *how to effectively manage a dApp in cloud environments with a decentralized control paradigm?* To answer this main research question, we analyze the requirements for a dApp control framework and conclude that control architecture and governance architecture are responsible for meeting decentralized requirements. In addition, coordination of decentralized control actions, including significant aspects of the consensus between peers and how to incentive collaboration between different peers, need to be investigated [14, 15].

2 Decentralized service control framework

In the literature, a variety of frameworks [6, 7, 10] for assessing the quality of services offered by different software producing organizations has been introduced. However, a control framework requires a peer-to-peer (p2p) control architecture [1] and an on-chain governance architecture [11] to control shared cloud infrastructures and maintain a high Quality of Service (QoS) of a dApp. In addition, a collaborative and proactive network in the control framework is needed. This study introduces a decentralized service control framework (DSConf) and elaborates on its architecture and constituent components.

2.1 Architecture design

Operator agent, decentralized control consensus, and dApp/service control agent are the main components of the DSConf (see Figure 1). The operator agent represents the operator of each organization in a dApp network, and it invokes the control layers which define different decentralized control patterns within the decentralized service control agent. Multiple control layers can be considered in a control architecture [16]. The lower-level control layer directly controls infrastructures/devices. The higher-level control layer is comprised of proposals and followed by a reply to execute or not (remote operators service operation) [13]. We also sub-divided service invocations into two categories: local dApp service invocations and remote dApp service invocations. Local dApp service invocations are performed by local services and consist of two control patterns: 1) operations on local service invoked by a local operator; 2) operations on local service invoked by remote operators. The Infrastructure as Code (IaC) service is invoked by logic defined on-chain in the first control pattern. The operator can call a local service function, and the function invokes the IaC service, which changes the state of the concerned infrastructure. In the second control pattern, a remote operator can propose a local dApp service invocation to one of the participating operators in the network. Remote dApp service invocations are performed on remote services (e.g., AWS). There are also two remote control patterns: 1) operations on remote services invoked by a local operator; 2) operations on remote services. In the first control pattern, an operator invokes a remote service. The second control pattern is a proposal sent to a global operator to invoke

a remote service. We provide a global coordination mechanism for decentralized control consensus to achieve on-chain governance in a p2p network. With the mechanism, an operator can vote on a particular proposal that affects the whole network. **Decentralized service control agent -** In the decentralized

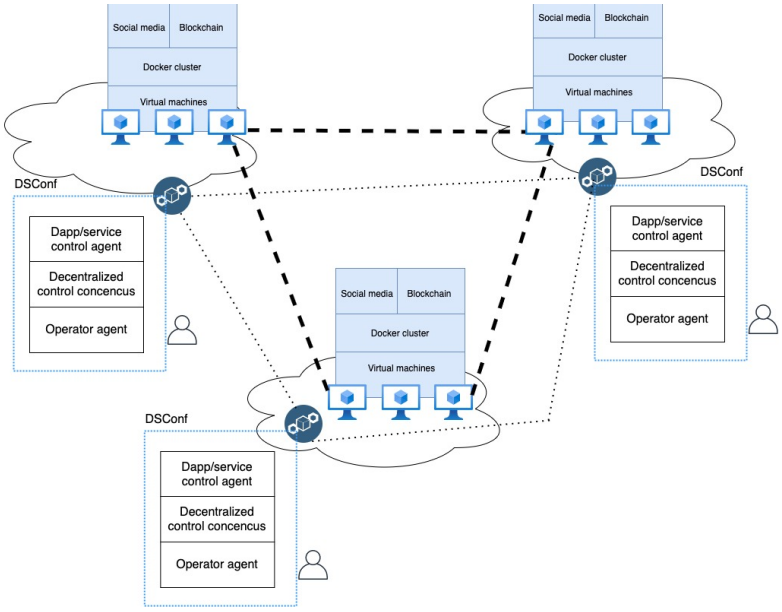

Fig. 1. Architecture design DSConf

service control agent, the network operators can execute and propose different control patterns. The service operations can be invoked locally and remotely. This decentralized service control agent uses the Co-util protocol to incentivize operators to list/verify these service invocations and propose/execute them.

Local dApp service invocation can be performed without a proposal to DSConf and can act quickly if the QoS declines and concerns the operator's part of the shared infrastructures. The operator can invoke local services that invoke the IaC service (scale VM, migrate services, etc.), which will change the state of the infrastructure. Local dApp service invocations by *remote operators* make use of a proposed mechanism. We suppose an operator from organization X detects a decline in the QoS of the dApp caused by the infrastructure that organization Y provided. In that case, the operator from organization X can propose specific local dApp service invocations to organization Y. The operator from organization Y can accept/decline this proposal. By accepting the proposal, the IaC service will be invoked by the on-chain logic of the local service. **Remote dApp service invocation** is a possibility that an operator performs a remote dApp service invocation. For example, a service invocation on the AWS platform

is not implemented locally. After executing this service invocation, the operator can list the performed remote dApp service invocation. The remote dApp service invocation will be verifiable. The incentive for listing these invocations will be explained in section 2.1. Remote dApp service invocations by remote operators make use of *remote operators (proposal)*. For instance, an operator from organization X can send a proposal to organization Y to perform a remote dApp service invocation. The operator replies to this request and performs the specific operational invocation on a remote service. This reply triggers the operational invocation to get automatically listed to be verified by another operator. The following types of the remote dApp service invocation by remote operators: *Propose* remote dApp service invocation. Define: proposal id, organization id that proposes, organization id proposed to, service operation, description, remote dApp service invocation bit (e.q. operator A proposes a remote dApp service invocation to operator B). *Reply* to a remote dApp service invocation proposal. Define: proposal id, reply (e.q. Operator B replies to the remote dApp service invocation proposal. On agreement, the proposal gets listed to be verified by another operator. Operator B's service invocation uses a different system (e.q. AWS dashboard)). *Verify* listed service invocation. The *collaboration protocol* supports the decentralized service control agent to create a collaborative p2p network of operators. Often peers in a network need an incentive to help others. Our framework implemented a protocol that lets operators gain voting power by helping others. Moreover, the voting power can be used in the global coordination mechanism. The collaboration protocol exists in two parts: DSConf introduces non-transferable tokens representing the network's voting power. The *non-transferable token* represents an operator's voting power within the network. It can be used in the weighted voting of the global DSConf proposal mechanism. An operator receives an initial amount of non-transferable tokens by joining the network: I_{amount}. The *Co-Util protocol* is introduced to enforce the operators to collaborate by increasing the operator's utility when helping other operators instead of displaying selfish behavior [4]. The action operators can take and the corresponding pay-off. Following the pay-off matrix, operators get rewarded for proposing, invoking, and verifying service invocations: the different operational control patterns and the associated rewards/penalties.

2.2 Decentralized control consensus

An on-chain voting mechanism can enable the consortium to change/maintain the DSConf network. The global proposal module consists of four functions: *Create* a vote, which is an index, title, description, creator, timestamp, duration, list of operators that agree, list of operators that disagree, answer if is passed or did not pass. *Reply* on a vote, which is an index of the vote someone wants to reply to, the reply (agree/disagree). *Close* a vote, which is an index of the vote someone wants to close. *Get* all votes.

In addition, vote options could be about the following topics: 1) creating new control functions; 2) blocking operators; 3) onboarding operators.

Voting incentivization protocol There is a need for a voting incentivization protocol to incentivize operators to behave pro-actively in these votes. The voting mechanism will use non-transferable tokens that represent the voting power of an operator. This voting power can be used in global proposals. Every peer in the DSConf network will get an initial amount of I_{amount} by joining the network. The stakes will be evenly distributed in the initial state of the DSConf network [5] [9]. A Proof of Stake(PoS) mechanism will be used to incentivize operators to vote on these proposals because choices often have to be made quickly. There has to be an incentive for the operators to reply as soon as possible. A debating period can be set for a vote in the voting process. When someone responds within the debating period, her new stake will be $C_{amount} + REWARD$. Where C_{amount} is the current amount of voting power the operator posses. If the participant of the DSConf did not manage to vote within this period, the penalty would be that they get slashed. This means that the new amount of their stake will be $C_{amount} - PENALTY$ [8]. *Development proposal.* The DSConf network could deploy new functionalities, update current functionalities, or change existing governance mechanisms. One of the participating organizations/operators can negotiate an off-chain deal to hire a developer. After that, the operator can vote to decide if the DSConf agrees with the proposal and the negotiated terms. If the DSConf agrees, everybody has to deposit the funds off-chain to the operator that arranged the proposal. Granted that everybody deposits, the operators confirm the proposal with the developer. *Block/unblock malicious operator.* A vote could be about blocking malicious operators. If the malicious activity is monitored, an operator can invoke a vote, report an operator, and describe the violation. If the vote passes, the operator is blocked for a certain amount of time. *On-boarding proposal.* The proposal could also be about onboarding an organization. There can be a vote to decide if a new organization can be added to the DSConf network.

3 Experiments

The purpose of our experiments is to verify the key performance indicators of the DSConf. These indicators are 1) usability, 2) reliability, 3) time-critical performance. The transactions to the DSConf chaincode contracts will be executed and captured in a shell environment. We observe the operator states as the output of the chaincode invocation for each control pattern.

Usability - In this section, the usability of DSConf is tested. We test the operational action contracts, and there are four control patterns and a voting mechanism that need to be tested. After every experiment of a control pattern, the ledger is re-initiated and the contracts redeployed, so the results are more straightforward to understand. *Local dApp service invocation by local operator*, in which the operator Org1MSP invokes a local service operation, and it does not need approval from others to invoke this operation. After executing the invocation functions, we can retrieve the different operators to see if the reward

of +2 is assigned to Org1MSP. In table 1, we can see that the reward is assigned correctly.

Table 1. Operators state with different control functions

Control pattern	OclToken			Reply
	Org1MSP	Org2MSP	Org3MSP	
Local-Local	202	200	200	-
Local-Remote	202	201	200	Agree
	199	200.5	200	Disagree
Remote-Local	201	202	200	-
Remote-Remote	201	200	200	Agree
	201	202	201	Disagree
Global	200	200	200	-

Reliability - The reliability of the DSConf network can be tested by experimenting with the disruption of the network. Experiments regarding the reliability of the service invocations are performed by executing the different functions implemented in the Global DSConf voting contract. Important to mention is the V_{min} variable that is set to 60. This means that 60% of the organization has to reply on a vote to pass. Org2MSP will be the organization that is offline in this experiment. Subsequently, Org1MSP will create a vote and immediately reply to this vote. After that, Org3MSP will reply and close the vote. The reliability results show that if a disturbance in the network arises, in this case, one organization node that is unreachable in a network of three organization nodes, the network still functions. The decentralized control functions in the network are still operational, and control invocations can still be proposed to the unreachable node. However, the unreachable node first has to be reachable again to invoke control actions.

Table 2. Time spend of service invocation

Execution time - Milliseconds(MS)			
Functions	Average(10x)	Fastest	Slowest
Action proposal	2.24	1.51	3.33
Execute action by proposal	16.31	10.73	29.83
Execute action without proposal	15.2516	10.48	25.12
List action	3.38	1.87	5.34
Reply proposal	16.18	14.21	20.11

Time critical performance In the performance experiments, the execution time of the different functions is measured. These measurements of the DSConf are essential because of the time-critical nature of certain service invocations. The execution measurement is performed from the beginning of the chaincode call until the end of the function executions, returning the return value and execution time. Another critical aspect of the architecture is that functions within the DSConf network are sequential. We measure the execution performance of the chaincode functions. Every function is executed ten times to gather enough data.

We evaluate the performance of *service invocation* and the *global DSConf voting mechanism*. The ' Execute action ' function is the most crucial function to measure execution time. This function has to perform time-critical operations on the concerned infrastructure.

Table 3. Time spend of voting mechanism

Execution time - Milliseconds(MS)			
Functions	Average(10x)	Fastest	Slowest
Create vote	2.66	1.24	5.18
Reply vote	3.06	2.39	4.94
Close vote	6.42	5.81	7.93

In these experiments, the execution time of the different functions was measured. The essential functions to measure were the local dApp service invocation functions. In table 2, these functions take around 16ms to execute, which is within a second. However, the execution time does not include the invocation of the IaC service. The other DSConf functions lie within an execution time of 2 to 30ms, also within a second. In table 3, we can see that the execution time of each function in the voting mechanism is under 10ms. So the overall performance of this experiment seems good.

4 Conclusion

In this research, we focus on how to effectively manage a dApp in cloud environments using a decentralized control paradigm. To conclude this question, we provide a review of the decentralized control framework at first. The review concludes that a p2p network and on-chain governance are essential for a decentralized control framework. In addition, there is a need for an incentive to enforce collaboration between operators in the p2p network.

To coordinate the decentralized control action within a p2p network and achieve on-chain governance, we provide a decentralized service control framework DSConf. In DSConf, a decentralized service control agent that uses a collaboration protocol to incentivize operators to collaborate is provided. In addition, a consensus voting mechanism enables mutual coordination between the operators in the DSConf network. This mechanism also uses an incentivization protocol to enforce pro-active behavior.

Acknowledgment

This work has been partially funded by the European Union's Horizon 2020 research and innovation programme by the ENVRI-FAIR (824068), BLUECLOUD (862409), and ARTICONF(825134) and by the LifeWatch ERIC.

References

1. Almasalma, H., Engels, J., Deconinck, G.: Peer-to-peer control of microgrids. In: 8th IEEE Benelux Young researchers symp. in Electrical Power Engineering (2016)
2. Bergers, J., Shi, Z., Korsmit, K., Zhao, Z.: Dwh-dim: A blockchain based decentralized integrity verification model for data warehouses. In: 2021 IEEE International Conference on Blockchain (Blockchain). pp. 221–228 (2021). https://doi.org/10.1109/Blockchain53845.2021.00037
3. Dhillon, V., Metcalf, D., Hooper, M.: Blockchain enabled applications. Berkeley, CA: Apress (2017)
4. Domingo-Ferrer, J., Martínez, S., Sánchez, D., Soria-Comas, J.: Co-utility: self-enforcing protocols for the mutual benefit of participants. Engineering Applications of Artificial Intelligence **59**, 148–158 (2017)
5. Fan, X., Chai, Q., Zhong, Z.: Multav: A multi-chain token backed voting framework for decentralized blockchain governance. In: International Conference on Blockchain. pp. 33–47. Springer (2020)
6. Farshidi, S., Jansen, S.: A decision support system for pattern-driven software architecture. In: ECSA. pp. 68–81. Springer (2020)
7. Farshidi, S., Jansen, S., Fortuin, S.: Model-driven development platform selection: four industry case studies. Software and Systems Modeling **20**(5), 1525–1551 (2021)
8. Leonardos, S., Reijsbergen, D., Piliouras, G.: Weighted voting on the blockchain: Improving consensus in proof of stake protocols. International Journal of Network Management **30**(5), e2093 (2020)
9. Nguyen, C., Dinh Thai, H., Nguyen, D., Niyato, D., Nguyen, H., Dutkiewicz, E.: Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. vol. PP, pp. 1–1 (06 2019). https://doi.org/10.1109/ACCESS.2019.2925010
10. Poon, L., Farshidi, S., Li, N., Zhao, Z.: Unsupervised anomaly detection in data quality control. In: 2021 IEEE International Conference on Big Data (Big Data). pp. 2327–2336. IEEE (2021)
11. Reijers, W., Wuisman, I., Mannan, M., De Filippi, P., Wray, C., Rae-Looi, V., Vélez, A.C., Orgad, L.: Now the code runs itself: On-chain and off-chain governance of blockchain technologies. Topoi pp. 1–11 (2018)
12. Saurabh, N., Rubia, C., Palanisamy, A., Koulouzis, S., Sefidanoski, M., Chakravorty, A., Zhao, Z., Karadimce, A., Prodan, R.: The ARTICONF Approach to Decentralized Car-Sharing. Blockchain: Research and Applications pp. 1–37 (may 2021). https://doi.org/10.1016/j.bcra.2021.100013
13. Stanciu, A.: Blockchain based distributed control system for edge computing. In: 2017 21st International Conference on Control Systems and Computer Science (CSCS). pp. 667–671 (2017). https://doi.org/10.1109/CSCS.2017.102
14. Tran, H., Hitchens, M., Varadharajan, V., Watters, P.: A trust based access control framework for p2p file-sharing systems. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences. pp. 302c–302c. IEEE (2005)
15. Wang, Y., Nguyen, T.L., Xu, Y., Tran, Q.T., Caire, R.: Peer-to-peer control for networked microgrids: multi-layer and multi-agent architecture design. IEEE Transactions on Smart Grid **11**(6), 4688–4699 (2020)
16. Wang, Y., Nguyen, T.L., Xu, Y., Tran, Q.T., Caire, R.: Peer-to-peer control for networked microgrids: Multi-layer and multi-agent architecture design. vol. 11, pp. 4688–4699 (2020). https://doi.org/10.1109/TSG.2020.3006883
17. Wu, K., Ma, Y., Huang, G., Liu, X.: A first look at blockchain-based decentralized applications. Software: Practice and Experience (2019)