financial application use cases have been considered during the design of the privacy solution of Quorum. From various (simple, high throughput) payment applications to derivative and loan use cases.

It's important to understand the Quorum privacy design does impose constraints on the application design. For example, updating a public contract state from a private contract is not permitted. Also, a private contract may not create a public contract. In general, private transactions cannot change public state, doing so would break the block validation/consensus algorithm.

Nevertheless, these constraints have not been unsurmountable in finding an an application/smart contract design for the financial use cases considered to date.

6.1 Payments

Various payment type use cases are useful proxies for many other financial applications. The simplest form of payment use case does not even require the use of smart contracts. The consenus critical information can be represented entirely in the data of the transaction itself and the blockchain of transactions.

Other types of payment applications may choose to use smart contracts to keep track of balances in lines of credit. Having the flexibility to manage private state consensus at the application layer is a powerful tool to help find designs to solve for the functionality and privacy requirements of many use cases. For example, consider an inter-bank payment use case where a payment from an account at Bank A is made to an account at Bank B. Bank C is not party to the payment and should not be able to know any information associated with this payment. It is the goal of the application design to insure that sufficient balance exists in accounts to cover the payment.

One particular design calls for the creation of a 'Bank' contract for each bank as illustrated in figure 4. In this design each Quorum node in the network would have 3 contracts, one for each Bank A, B, and C. In addition, a regulator node would be part of the network and would be party to all payment transactions. It is a further design requirement that payment transfers between accounts and banks be atomic operations, i.e. a single transaction must update both Bank contracts, all or nothing. In this design, the state of contract Bank A would only be in consensus with the Regulator node (and other Bank A nodes). The Bank A contract on Bank A's node would see all debits and credits between itself and all the Banks in the network. From Bank B's perspective, the state of Contract Bank A would only reflect the debits and credits between Bank A and Bank B, it would not know of any payments between Bank A and Bank C. In this design, the application would seek to confirm state consensus of its own Bank contract with only the regulatory node. By design, the state of the same contract on different Bank nodes would not be in state consensus.