Architecture of the Hyperledger Blockchain Fabric[*]

Christian Cachin

IBM Research - Zurich
CH-8803 Rüschlikon, Switzerland

cca@zurich.ibm.com

July 2016

Overview. A blockchain is best understood in the model of state-machine replication [8], where a service maintains some state and clients invoke operations that transform the state and generate outputs. A blockchain emulates a "trusted" computing service through a distributed protocol, run by nodes connected over the Internet. The service represents or creates an asset, in which all nodes have some stake. The nodes share the common goal of running the service but do not necessarily trust each other for more. In a "permissionless" blockchain such as the one underlying the Bitcoin cryptocurrency, anyone can operate a node and participate through spending CPU cycles and demonstrating a "proof-of-work." On the other hand, blockchains in the "permissioned" model control who participates in validation and in the protocol; these nodes typically have established identities and form a *consortium*. A report of Swanson compares the two models [9].

Hyperledger. The Hyperledger Project (`www.hyperledger.org`) is a collaborative effort to create an enterprise-grade, open-source distributed ledger framework and code base. It aims to advance blockchain technology by identifying and realizing a cross-industry open standard platform for distributed ledgers, which can transform the way business transactions are conducted globally. Established as a project of the Linux Foundation in early 2016, the Hyperledger Project currently has more than 50 members.

Hyperledger Fabric. *Hyperledger Fabric* (`github.com/hyperledger/fabric`) is an implementation of a distributed ledger platform for running smart contracts, leveraging familiar and proven technologies, with a modular architecture allowing pluggable implementations of various functions. It is one of multiple projects currently in *incubation* under the Hyperledger Project. A developer-preview of the Hyperledger Fabric (called "v0.5-developer-preview") has been released in June 2016 (`github.com/hyperledger/fabric/wiki/Fabric-Releases`).

The distributed ledger protocol of the fabric is run by *peers*. The fabric distinguishes between two kinds of peers: A *validating peer* is a node on the network responsible for running consensus,

[*]This text represents the view of the author and is directly based on the current Hyperledger Fabric documentation and code, available online at `github.com/hyperledger/fabric/`.

The author is grateful to many collaborators at IBM and in the Hyperledger Project, including Sheehan Anderson, Elli Androulaki, Angelo De Caro, Konstantinos Christidis, Jason Yellick, Chet Murthy, Binh Nguyen, Michael Osborne, Simon Schubert, Gari Singh, Alessandro Sorniotti, Marko Vukolić, Sharon Weed, and John Wolpert.

validating transactions, and maintaining the ledger. On the other hand, a *non-validating peer* is a node that functions as a proxy to connect clients (issuing transactions) to validating peers. A non-validating peer does not execute transactions but it may verify them.

Some key features of the current fabric release are:

- A permissioned blockchain with immediate finality;
- Runs arbitrary smart contracts (called *chaincode*) implemented in Go (`golang.org`):
  - User-defined chaincode is encapsulated in a Docker container;
  - System chaincode runs in the same process as the peer;
- Consensus protocol is pluggable, currently an implementation of Byzantine fault-tolerant consensus using the PBFT protocol [4] is supported, a prototype of SIEVE [3] to address non-deterministic chaincode is available, and a protocol stub (named NOOPS) serves for development on a single node;
- Security support through certificate authorities (CAs) for *TLS certificates*, *enrollment certificates*, and *transaction certificates*;
- *Persistent state* using a key-value store interface, backed by RocksDB (`rocksdb.org`);
- An event framework that supports pre-defined and custom events;
- A client SDK (Node.js) to interface with the fabric;
- Support for basic REST APIs and CLIs.

Support for non-validating peers is minimal in the developer preview release.

Architecture. The validating peers run a BFT consensus protocol for executing a replicated state machine that accepts three types of *transactions* as operations:

Deploy transaction: Takes a chaincode (representing a smart contract) written in Go as a parameter; the chaincode is installed on the peers and ready to be invoked.

Invoke transaction: Invokes a transaction of a particular chaincode that has been installed earlier through a deploy transaction; the arguments are specific to the type of transaction; the chaincode executes the transaction, may read and write entries in its state accordingly, and indicates whether it succeeded or failed.

Query transaction: Returns an entry of the state directly from reading the peer's persistent state; this may not ensure linearizability.

Each chaincode may define its own persistent entries in the state. The blockchain's hash chain is computed over the executed transactions and the resulting persistent state.

Validation of transactions occurs through the replicated execution of the chaincode and given the fault assumption underlying BFT consensus, i.e., that among the n validating peers at most $f < n/3$ may "lie" and behave arbitrarily, but all others execute the chaincode correctly. When executed on top of PBFT consensus, it is important that chaincode transactions are deterministic, otherwise the state of the peers might diverge. A modular solution to filter out non-deterministic transactions that are demonstrably diverging is available and has been implemented in the SIEVE protocol [3].

Membership among the validating nodes running BFT consensus is currently static and the setup requires manual intervention. Support for dynamically changing the set of nodes running consensus is planned for a future version.

As the fabric implements a permissioned ledger, it contains a security infrastructure for authentication and authorization. It supports *enrollment* and *transaction authorization* through public-key certificates, and *confidentiality* for chaincode realized through in-band encryption.

More precisely, for connecting to the network every peer needs to obtain an *enrollment certificate* from an *enrollment CA* that is part of the membership services. It authorizes a peer to connect to the network and to acquire *transaction certificates*, which are needed to submit transactions. Transaction certificates are issued by a *transaction CA* and support pseudonymous authorization for the peers submitting transactions, in the sense that multiple transaction certificates issued to the same peer (that is, to the same enrollment certificate) cannot be linked with each other.

Confidentiality for chaincodes and state is provided through symmetric-key encryption of transactions and states with a blockchain-specific key that is available to all peers with an enrollment certificate for the blockchain. Extending the encryption mechanisms towards more fine-grained confidentiality for transactions and state entries is planned for a future version.

Discussion. Consensus protocols for blockchain are currently being debated very actively, in research [6, 10] but also by fintech startup companies (e.g., `tendermint.com`, `kadena.io`). The fabric's design uses a modular notion of consensus, which is aligned with the well-established concept of consensus in distributed computing. This ensures that the blockchain-related features of the fabric can be developed independently of the specific consensus protocol. The PBFT protocol [4] has been implemented as the first one in the fabric because of its prominence: it benefits from the experience of almost 20 years of systems-level research on Byzantine consensus, is closely related to well-known protocols like viewstamped replication and Paxos [7], has been analyzed in many environments [5, 1], and is described in textbooks [2].

Conclusion. The Hyperledger Fabric is a permissioned blockchain platform aimed at business use. It is open-source and based on standards, runs user-defined smart contracts, supports strong security and identity features, and uses a modular architecture with pluggable consensus protocols.

The fabric is currently evolving and being actively developed under the governance of the Hyperledger Project. More information about the fabric is available online at
`github.com/hyperledger/fabric/blob/master/docs/protocol-spec.md`

References

[1] Y. Amir, B. A. Coan, J. Kirsch, and J. Lane. Prime: Byzantine replication under attack. *IEEE Transactions on Dependable and Secure Computing*, 8(4):564–577, 2011.

[2] C. Cachin, R. Guerraoui, and L. Rodrigues. *Introduction to Reliable and Secure Distributed Programming (Second Edition)*. Springer, 2011.

[3] C. Cachin, S. Schubert, and M. Vukolić. Non-determinism in Byzantine fault-tolerant replication. e-print, arXiv:1603.07351 [cs.DC], 2016. URL: `http://arxiv.org/abs/1603.07351`.

[4] M. Castro and B. Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, Nov. 2002.

[5] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti. Making Byzantine fault tolerant systems tolerate Byzantine faults. In *Proc. 6th Symp. Networked Systems Design and Implementation (NSDI)*, pages 153–168, 2009.

[6] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology: Eurocrypt 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015.

[7] B. Liskov. From viewstamped replication to Byzantine fault tolerance. In B. Charron-Bost, F. Pedone, and A. Schiper, editors, *Replication: Theory and Practice*, volume 5959 of *Lecture Notes in Computer Science*, pages 121–149. Springer, 2010.

[8] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, Dec. 1990.

[9] T. Swanson. Consensus-as-a-service: A brief report on the emergence of permissioned, distributed ledger systems. Report, available online, Apr. 2015. URL: `http://www.ofnumbers.com/wp-content/uploads/2015/04/Permissioned-distributed-ledgers.pdf`.

[10] M. Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *Open Problems in Network Security, Proc. IFIP WG 11.4 Workshop (iNetSec 2015)*, volume 9591 of *Lecture Notes in Computer Science*, pages 112–125. Springer, 2016.