
BUILDING
BLOCKCHAIN
APPS
MICHAEL JUNTAO YUAN
Foreword by MARC FLEURY, founder of JBoss and Two Prime

Building Blockchain Apps

require media files, databases, and other off-chain data to function. The dapp could use online services to store and manage data. Here are some examples:

- The IPFS (https://ipfs.io/) is a blockchain-based media file storage and exchange service protocol. Dapps can store large user files on IPFS and make them accessible everywhere.
- Swarm (https://ethersphere.github.io/swarm-home/) is a file storage and sharing solution built on top of Ethereum.
- GitHub (https://github.com/), Dropbox (https://www.dropbox.com/), or Google Drive (https://www.google.com/drive/) are examples of traditional Internet file storage and sharing services that can be accessible by individual dapp users. You can use GitHub or Dropbox web sites to serve dapp JavaScript files directly from an individual user's accounts.
- Database as a service (DBaaS) providers such as Microsoft Azure SQL (https://azure.microsoft.com/en-us/services/sql-database/), AWS Relational Database Service (https://aws.amazon.com/rds/), Google BigQuery (https://cloud.google.com/bigquery/), and MongoDB Atlas (https://www.mongodb.com/cloud/atlas) are examples of database services that can be utilized by dapps to store application data.
- An off-chain data service is a query interface to search and browse blockchain data, such as transactions, accounts, and smart contract function calls. It is potentially much more powerful, versatile, and scalable than calling the view functions to get smart contract data. This approach is discussed in more detail in Chapter 10.

A common design practice to ensure the safety and validity of off-chain data is to store the data's hash in on-chain smart contracts.

A dapp is more complex than most web applications. From the start, you need to design which part of the application is based on blockchain smart contracts, which part utilizes off-chain server-side data, and which part is the client-side UI. Each of those elements requires its own software stack to function and communicate with the rest of the application.

Dapp Showcases

Because of Ethereum's slow confirmation time (up to ten seconds) and high gas fees for executing smart contract functions, successful Ethereum dapps so far are all financial applications that do not require frequent user interactions.

Uniswap

One of the most polished Ethereum dapps is the Uniswap exchange. It is a decentralized crypto token exchange. The idea is that some people will make initial contributions to a liquidity pool (as market makers) and earn a share of the trading fees. All other traders will trade against the liquidity pool based on a simple pricing formula of supply and demand. If a token becomes scarce in the liquidity pool, its price against the ETH will increase, incentivizing holders to sell it back to the liquidity pool. This mechanism allows trading to happen in a completely automated manner

without matching for counter parties. The entire Uniswap system is a set of smart contracts on the Ethereum blockchain. The application state is completely stored in and managed by the contracts.

The Uniswap project has developed a polished UI (Figure 7.2) to interact with its underlying smart contracts. The UI is completely written in web3 JavaScript and is fully internationalized. Through the dapp UI, novice users can contribute to the liquidity pool and earn fees for their crypto deposits or can immediately start trading pairs of ERC20 tokens.

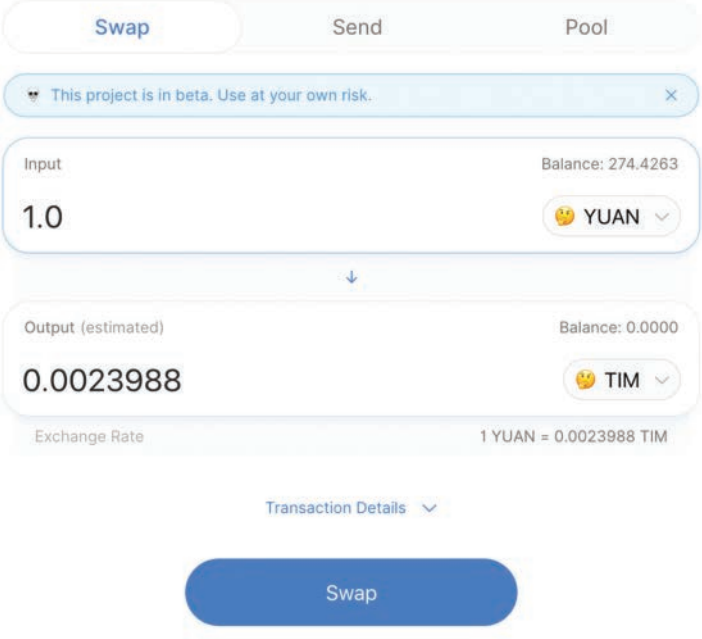

Figure 7.2 The Uniswap UI

The interesting aspect of the Uniswap dapp is that it is truly decentralized. All the application logic and its data are stored on the Ethereum blockchain. Anyone can create a web site to host the JavaScript dapp, and all those copies of dapps will behave the same way as they all get their logic and data from the blockchain. In Uniswap, Ethereum has truly become a "computer" back end.

CryptoKitties

The CryptoKitties game took over Ethereum by storm in late 2017. It gave rise to the idea of crypto collectibles and nonfungible tokens, which later become the ERC721 specification.

CryptoKitties are unique digital entities that exist on the Ethereum blockchain. They are data in smart contracts. Their uniqueness is guaranteed by the contract code. Each CryptoKittie has an associated owner address. CryptoKitties can then by bought, sold, and traded on the blockchain.

The interesting thing about CryptoKitties is that they are visually appealing (Figure 7.3). The dapp UI design visualizes the unique features of each digital entity. That contributed significantly to CryptoKitties' success.

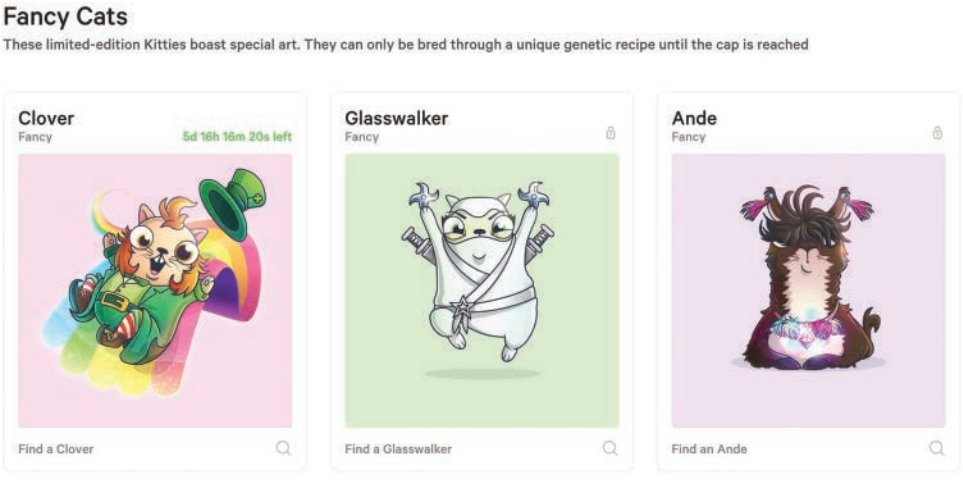

Figure 7.3 CryptoKitties

Gambling Games

Gambling games are popular dapps. They are great use cases for smart contracts, with transparent rules and betting pools. They also benefit greatly from largely unregulated cryptocurrencies.

However, gambling games are often interactive in nature, requiring users to bet often in response to other people's real-time bets. Ethereum's slow confirmation time represents a great barrier for this type of game. Faster blockchains such as EOS and Tron, which is itself a fork from Ethereum, are now blockchains specialized in gambling dapps.

Interactive Dapps

Most Internet applications are interactive. For Ethereum blockchains to support interactive applications, performance is an important factor. The Lity project creates high-performance extensions to the Ethereum protocol and tools. It enables us to create interesting interactive dapps. See Chapter 16 for several complete examples.

Conclusion

A dapp is typically a web3 application that runs in tandem with a wallet application. It interacts with smart contracts on the blockchain for essential data and core application logic. It can also use local storage or third-party services to store and manage nonessential data that is private to the user or could be regenerated by the public. In the next chapter, I will discuss how applications use blockchain data services, in addition to blockchain transactions, to provide a rich user experience.

This page intentionally left blank

8

Alternatives to Dapps

The concept of dapps is compelling and native to blockchain technology's most obvious use cases, such as peer-to-peer financing. However, the world is never binary. There are also many blockchain application use cases that could fit into the model of traditional web applications. Those are typically use cases where the public or consortium blockchain's transparency and immutability can add value to an existing business. The application only needs the blockchain as a feature and does not need to decentralize the entire application itself. A payment processor for e-commerce web sites to accept cryptocurrencies is a good example. A crypto asset exchange (crypto to crypto or crypto to fiat) is another example.

For those applications, we need to make blockchain transactions and/or make smart contract function calls from a server. To do that, the accounts' private keys or keystores and passwords must be managed on the server side.

- For new transactions, it is obvious since the sender account needs to use its private key to authenticate the funds transferring out of it.
- For modifying contract states, an Ethereum-compatible blockchain requires the party requesting the change to pay a "gas fee" for network maintainers (miners) to validate the request and record the change in new blocks. That also requires transferring funds (a gas fee) out of the requestor's account and hence requires its private key.

In this chapter, we explore how to access Ethereum-compatible blockchain functionalities from a web app. The basic approach is to use a web3-compatible library but without a client-side wallet like Metamask.

JavaScript

The node.js framework enables JavaScript applications to be deployed on the server side. It is hence conceivable to use the web3.js library (or the compatible web3-cmt.js library) in a node.js server application.