Typical architecture of a Dapp with a browser client

I am trying to understand the typical architecture of a Dapp with browser clients. Is the below understanding correct?

[Web Browser *(end user)***]** <==> **[Server** *(web application/javascript <==> web3 <==> ethereum client like geth)***]** <==> **[Ethereum Network** *(solidity code)***]**

dapp-development dapps dapp-design

edited Feb 16 at 10:50

asked Feb 15 at 11:44
Ashish Sinha
336 1 13

Hey, basically you are correct. Could you help me understand what you mean with ethereum client? – Borinho Feb 15 at 12:50

Hi Borinho, ethereum client like geth, eth, pyethapp, etc which can communicate with smart contracts on the ethereum network. – Ashish Sinha Feb 16 at 10:20

1 Answer

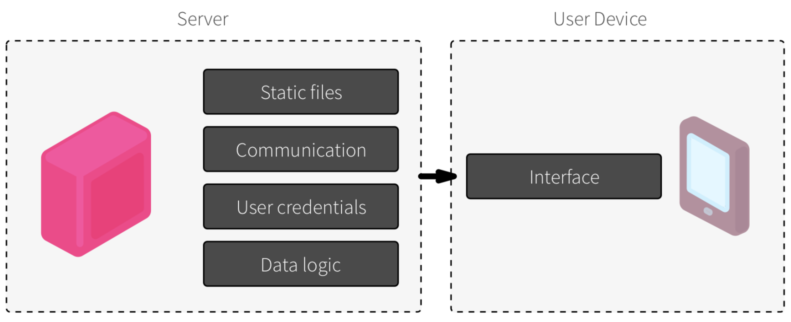
I'm not sure if you are describing the dataflow rather than the actual system architecture, but if you are it isn't entirely correct.

Overview of decentralised applications:

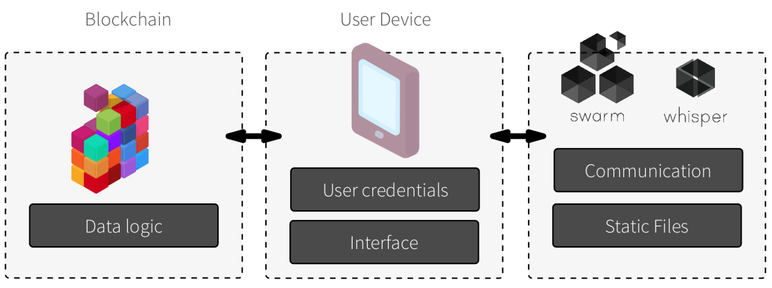
Alex Van de Sande wrote a great article on building serverless applications and in it he contrasts the distinction between a centralised build and a decentralised build.

https://blog.ethereum.org/2016/07/12/build-server-less-applications-mist/

Traditional Centralised Structure



New Generation Decentralised Structure



The article talks a lot about Mist, but this was when Mist really was the only shown in town. Now we have Parity and even MetaMask that allow users access to web3.js and a connection to the an Ethereum wallet, and importantly the ability to use the Ethereum wallet to sign transactions on the Ethereum blockchain/ network.

Considering your example:

[Web Browser *(end user)***]** <==> **[Server** *(web application/javascript <==> web3 <==> ethereum client)***]** <==> **[Ethereum Network** *(solidity code)***]**

The Ethereum client isn't usually on the server side, there are exceptions like the case of using something like MetaMask, but in that case you'd use MetaMasks serverside and not your own. The Ethereum client would be part of the web browser. Also the client is what communicates with the Ethereum Network, not your server. So your structure / relationship would need to be split into two parts and look more like:

Downloading the HTML and JS:

[Web Browser (end user <==> Ethereum client)**]** <==> **[Server** (web application/JavaJcript <==> web3)**]**

Interacting with the blockchain:

[Web Browser (end user <==> Ethereum client)**]** <==> **[Ethereum Network** (solidity code)**]**

edited Feb 16 at 11:22

answered Feb 15 at 13:09
Samuel Hawksby-Robinson
1,086 1 4 21

Above is a good answer. Just so it doesn't seem like server-side nodes is terribly unusual (nodejs, python, etc.) I would point to a similar question here: ethereum.stackexchange.com/questions/11624/… – Rob Hitchens Feb 16 at 6:32

Actually by ethereum client, I meant geth/eth/pyethapp. End users are general public who just type in www.xyz.com into the browser to access the app (dapp). We cannot expect them to have the entire geth installed on their laptop! There needs to be a web application on a server with server-side javascript (node.js), web3 and geth to help the web application to connect to the ethereum network. – Ashish Sinha Feb 16 at 10:47

Out of interest how are your users managing their Ethereum keys? – Samuel Hawksby-Robinson Feb 16 at 11:30

Hi Samyoul - I haven't yet gone till there. I am only trying to understand how a smart contract / dapp can have a web client. For example, when I visit etherscan.io, it communicates with the ethereum network (having smart contracts) to display the data it displays. How does it do it? Does the js on my local browser connect directly to the etheruem network? I don't think so. What I understand is that ethereum network can be accessed only through components like web3 (or similar) and geth (or similar) and I have none of these on my laptop. So where are these? Somewhere in between? – Ashish Sinha Feb 16 at 15:03

It depends on your use case. Do you want your users to interact with contracts? If so, the users will need a secure way to sign their transactions. A purely server side Ethereum client will mean that users won't be able to do this, unless you build an additional service that functions just like MetaMask. – Samuel Hawksby-Robinson Feb 16 at 15:11