AGORIC

Dean Tribble
November 2019
DeFi Hackathon

Hacking the Agoric Platform

Outline

- intro: brief Zoe
- autoswap demo: dapp UI page, wallet page
- zoom out to: local browser, webapp from dapp server, wallet webapp from local node, local swingset
- zoom out to: chain/dapp-server/client architecture diagram
- code walkthrough 1: dapp UI sends POST to dapp server, dapp server sends vat message to chain-side object to get current autoswap price, sends price down to dapp UI: shows Vats, not ERTP
- walkthrough 2: how dapp UI sends offer proposal through "bridge" to wallet
- ERTP
- Zoe interfaces
- lesson 3: walk through chain-side contract to do fixed price offer

What is a smart contract?

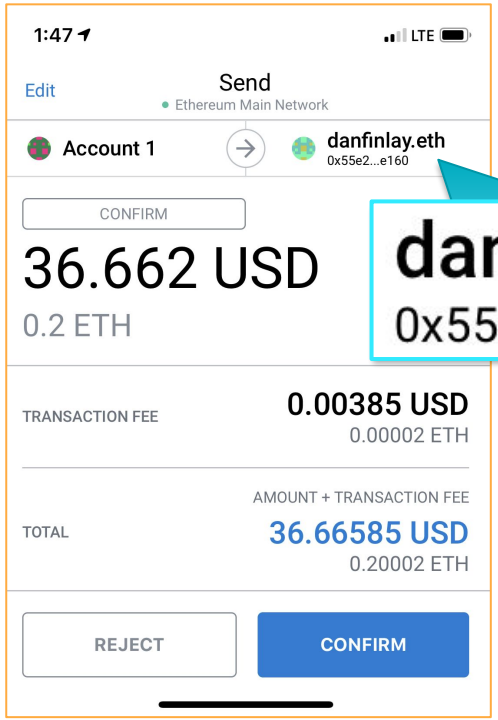
A contract-like arrangement, expressed in code, where the behavior of the program enforces the terms of the contract

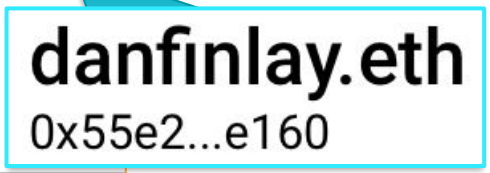
What does blockchain bring?

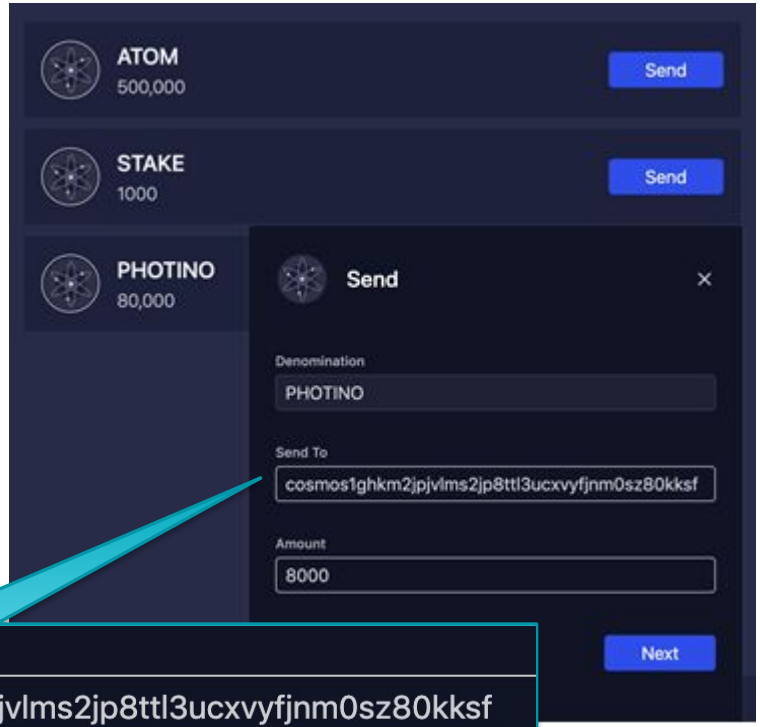
- Multiple **independent** computers …
  - Different **administration**
  - Different **jurisdictions**
- … vote to agree on …
  - **Data** to record
  - **Order** of events
  - Results of **computation**

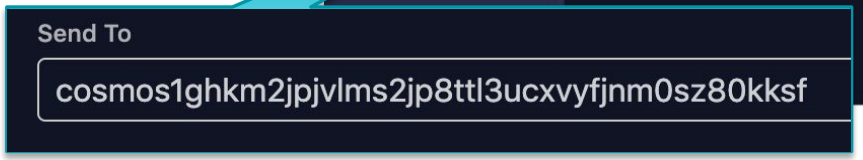
Computing with **Integrity**
a high-integrity “third party” for smart contracts

Low-level payments



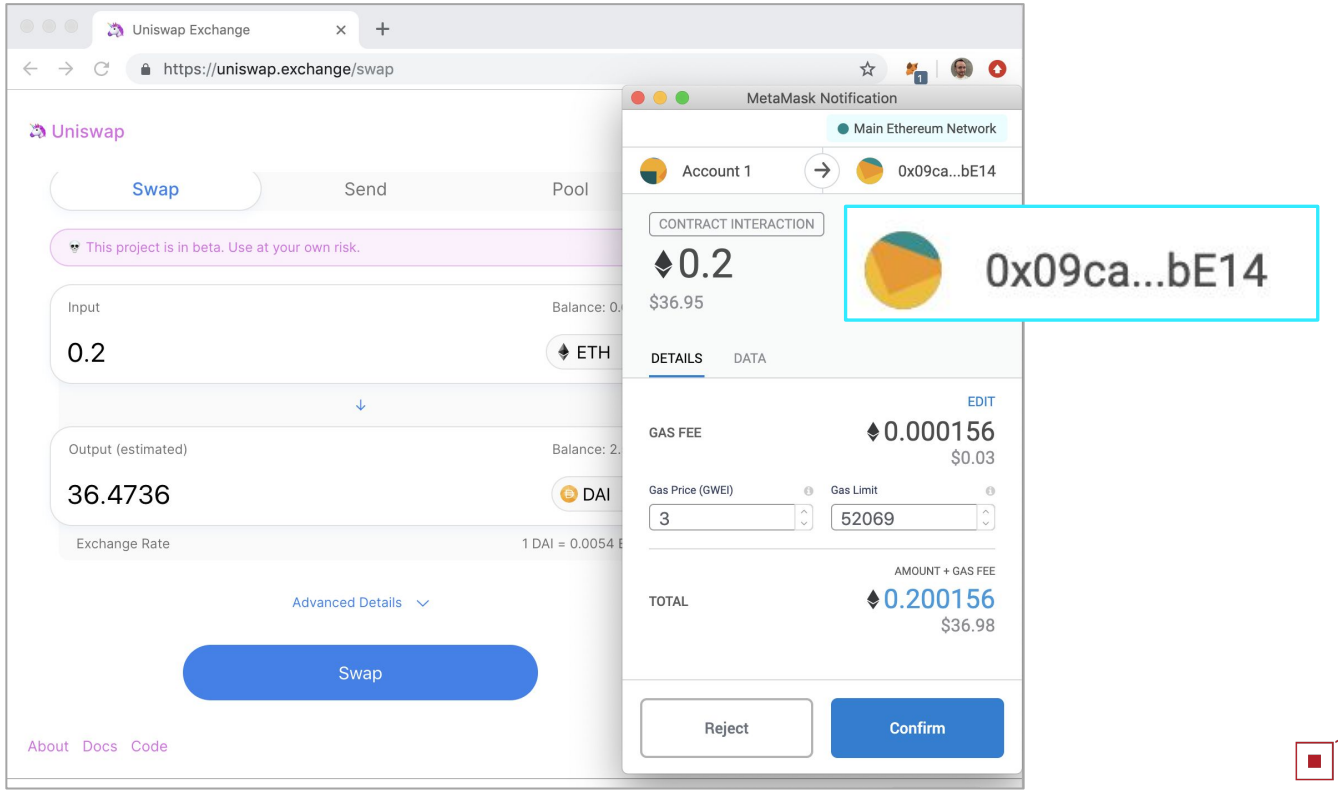






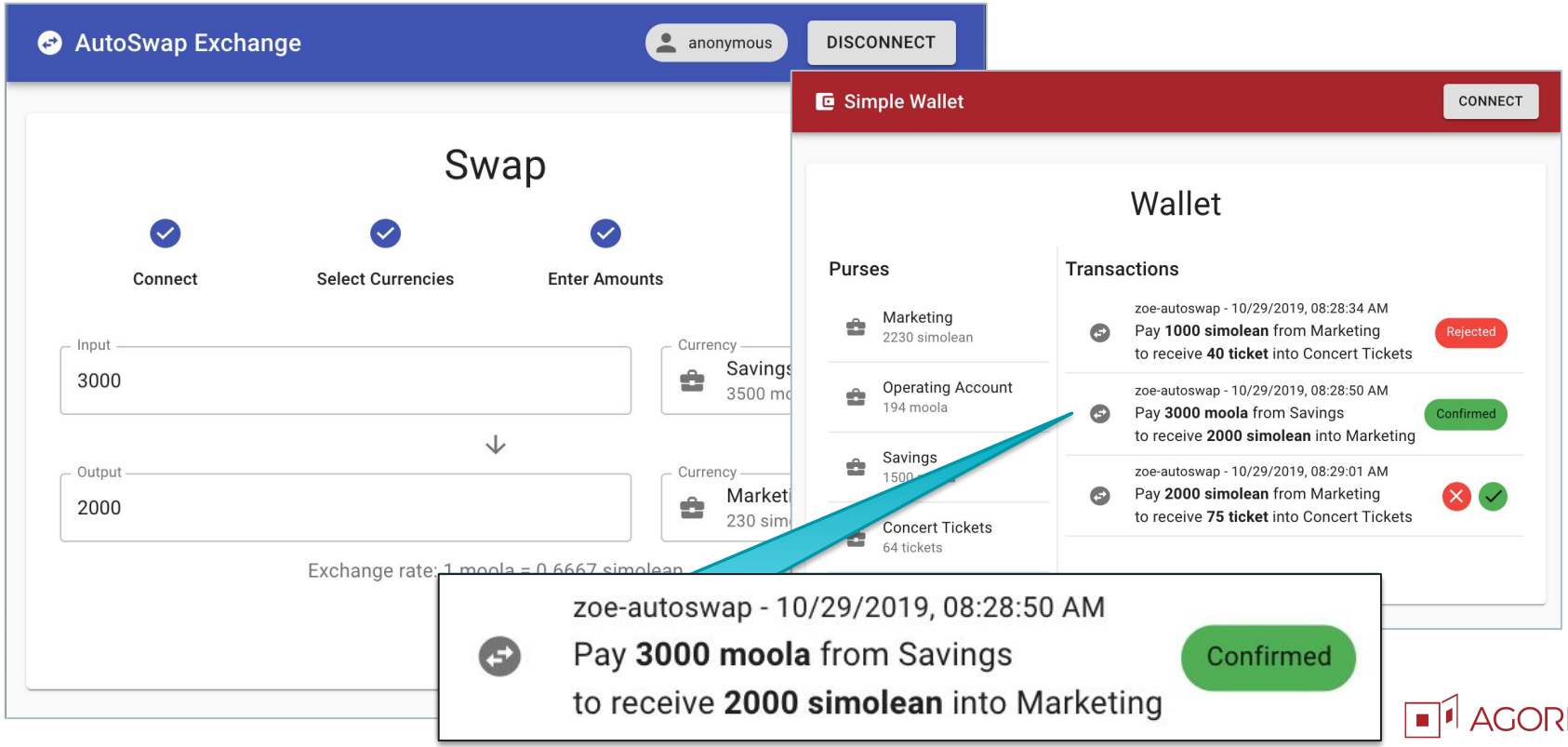
AGORIC

Low-level application UIs



AGORIC

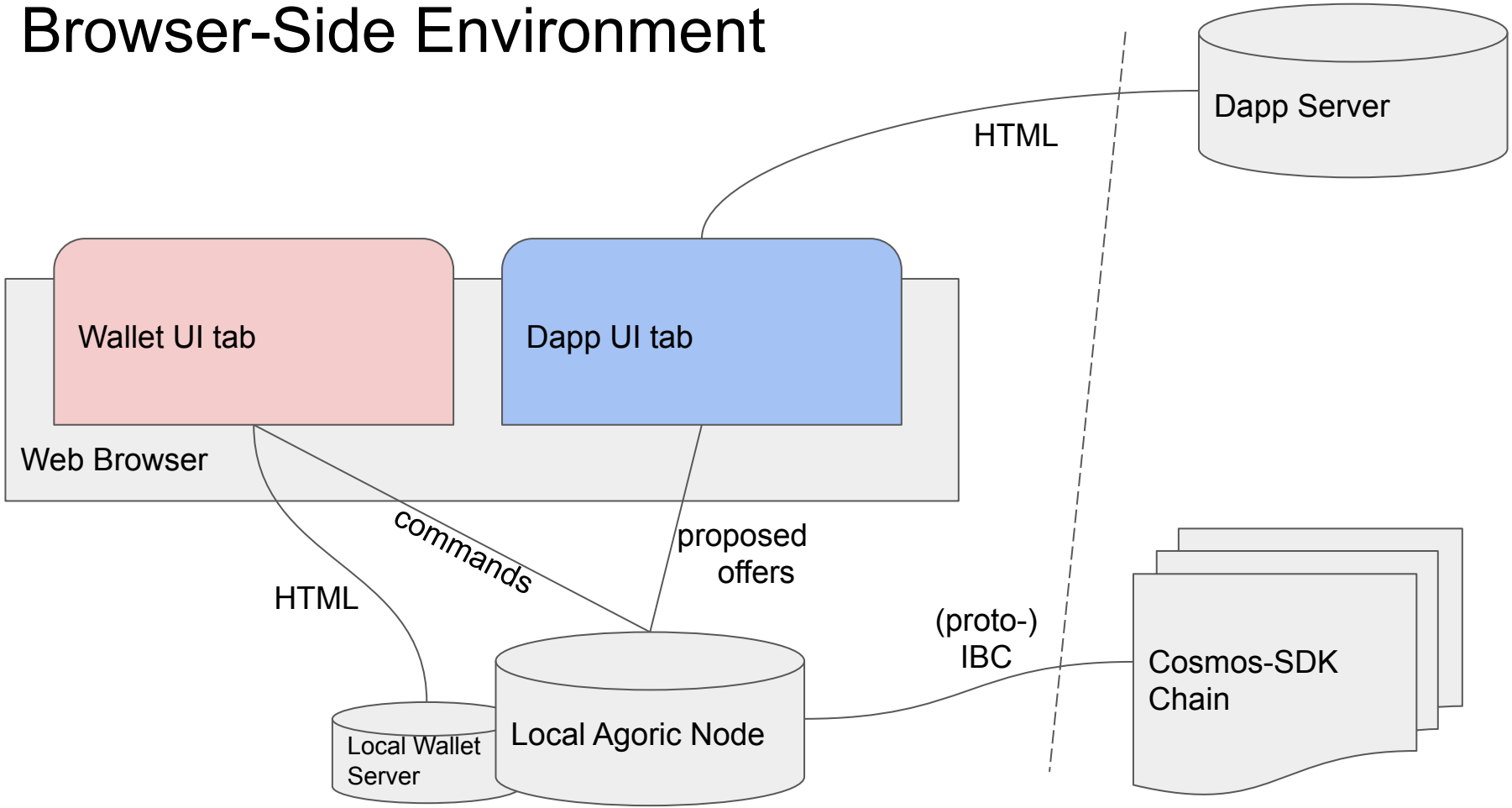
Contracts need Quid Pro Quo

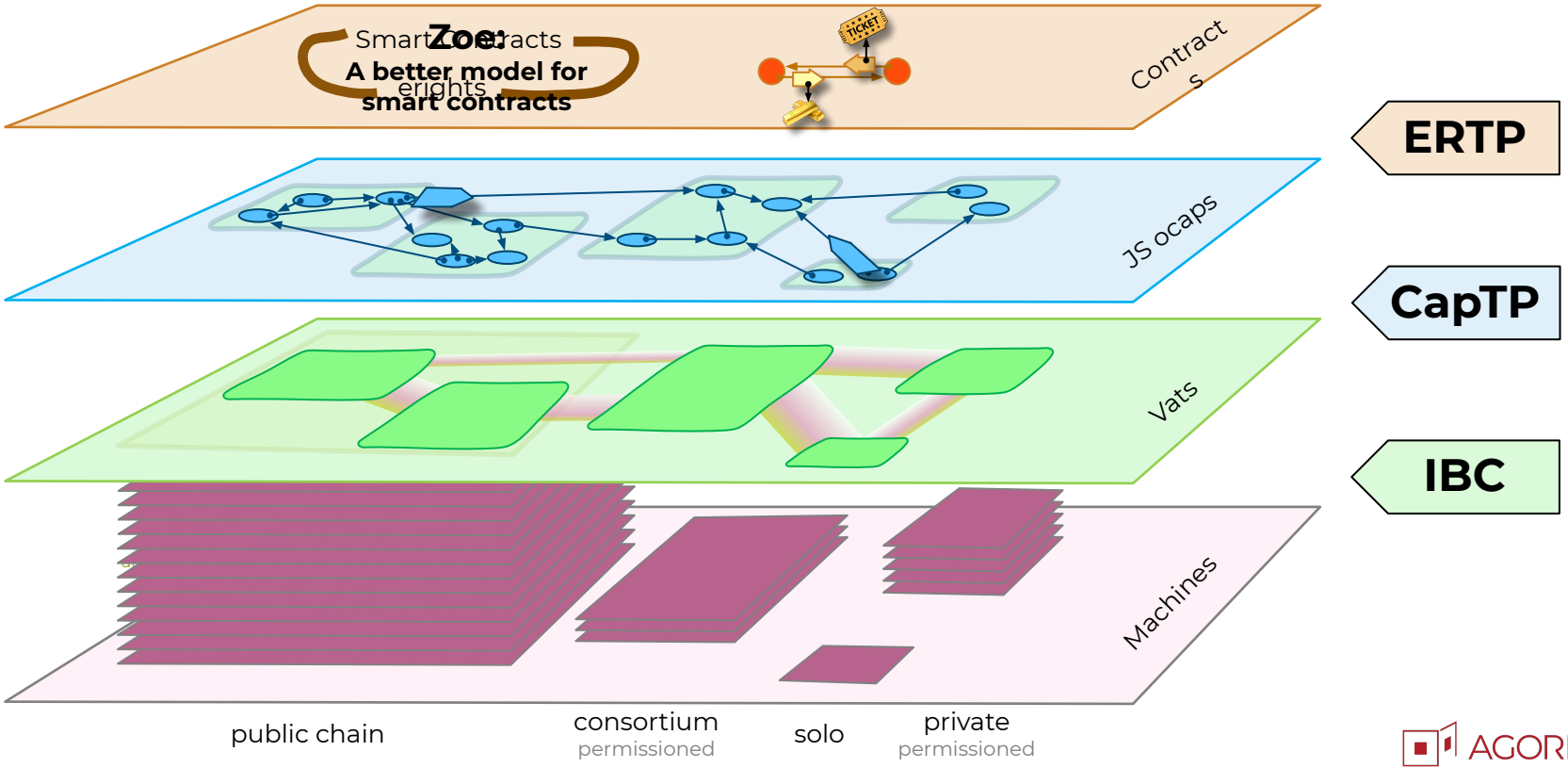


AGORIC

Under the hood

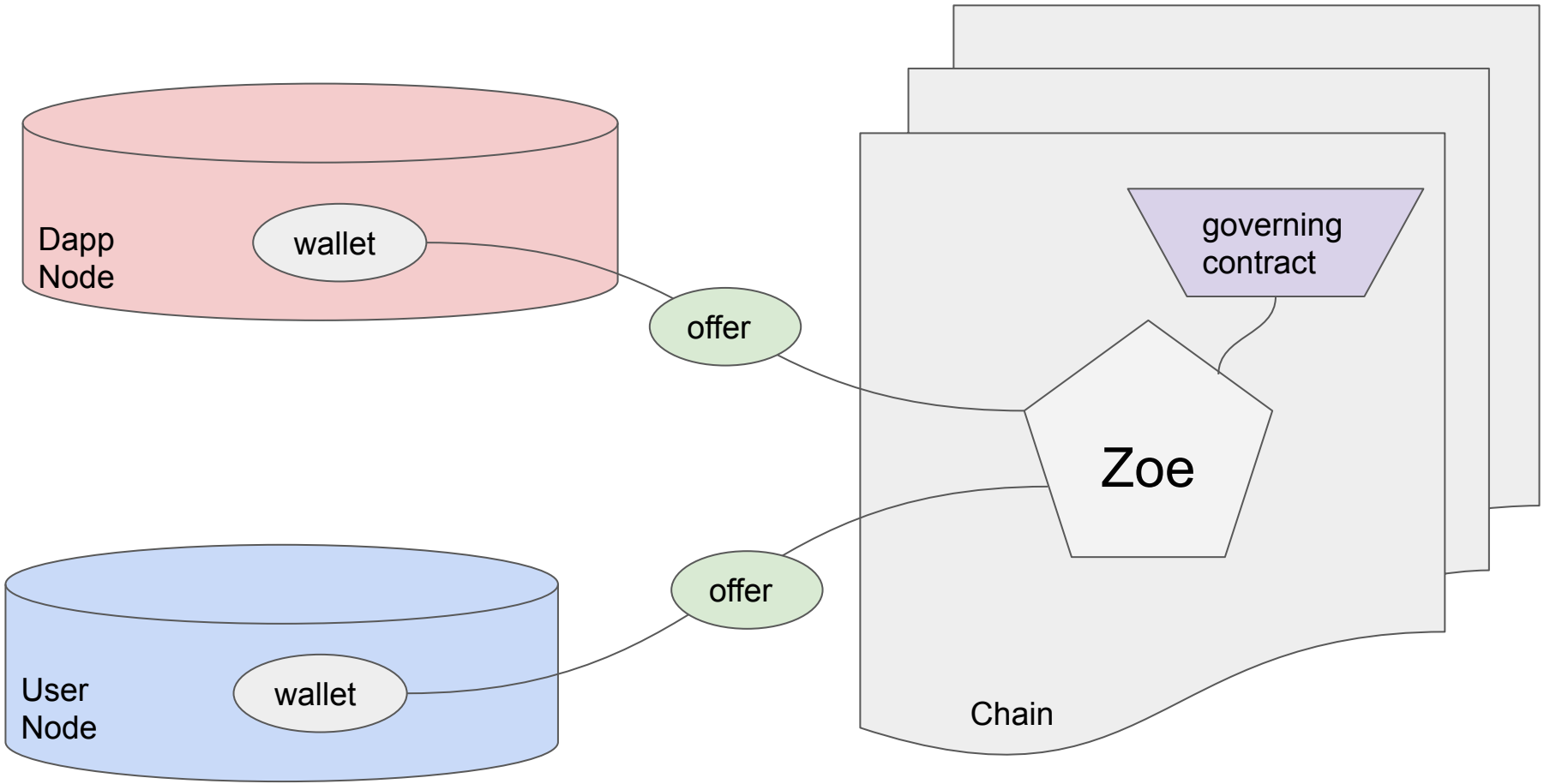
Browser-Side Environment




Smart contracts
Zoe:
A better model for
smart contracts
erights
TICKET
Contract
s
JS ocaps
Vats
Machines
ERTP
CapTP
IBC
public chain
consortium
permissioned
solo
private
permissioned

AGORIC

System Architecture



Code Walkthrough 1

- dapp ui sends request to dapp server
- "handler" in dapp server (vat code) gets request

```
chainContract~.getCurrentPrice().then(tellFrontend)
```

GetPrice https://github.com/Agoric/autoswap-frontend/blob/master/src/contexts/Application.jsx#L79

Code Walkthrough 1

- responds to dapp ui
- dapp ui updates element with
- current price

```
function messageHandler(message) {
    if (!message) return;
    const { type, data } = message;
    if (type === 'autoswapPrice') {
      dispatch(changeAmount(data, 1 - freeVariable));
    }
  }

  doFetch({
      type: 'autoswapGetPrice',
      data: {
        instanceId: CONTRACT_ID,
        extent0: inputAmount,
        assayId0: inputPurse.assayId,
        assayId1: outputPurse.assayId,
      },
    }).then(messageHandler);
```

Code Walkthrough 2

- dapp ui constructs proposed Offer
- sends proposed Offer through Bridge to wallet

```
const offerRules = harden({
  payoutRules: [ {
    kind: 'offerExactly',
    units: { assayId: assayId0, extent },
  },
  { kind: 'wantAtLeast',
    units: { assayId: assayId1 },
  },
  { kind: 'wantAtLeast',
    units: {},
  },
 ], exitRule: { kind: 'onDemand', },
});
```

```
return doFetch({
        type: 'walletAddOffer',
        data: {
          meta,
          offerRules: data,
        },
      }, true);
```

Code Walkthrough 2

- wallet uses wallet UI to present to user
- user approves
- wallet submits offer to contract

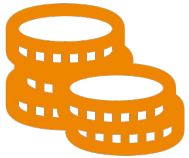
```
export function acceptOffer(state, date) {
  doFetch({
    type: 'walletAcceptOffer',
    data: date,
  }); // todo toast

  return state;
}
```

ERTP:
Electronic Rights Transfer Protocol

ERTP provides the foundation

PURSE — hold digital assets



PAYMENT — transfer digital assets

MINT — create digital assets

ASSAY — verify and claim assets

UNITS — describe the kind of assets

AGORIC

Mints and Payments

```
makeMint(name, config)
        ⇒ Mint

Mint
  mint(units)  ⇒ Payment

Payment
  getBalance() ⇒ Units
  getAssay()   ⇒ Assay
```

```
// setup for concert tickets
const ticketsMint =
    mints.makeMint("Concert", ticketConfig);

// publish concert ticket Assay
const venueA = ticketsMint.getAssay();

// create and return a ticket
return ticketsMint.mint(seats("E4"));

// send a new ticket to carol
const t = ticketsMint.mint(seats("E5"));
carol.receiveTicket(t);
```

Exclusive transfer with Assays and Purses

Assay

- **claimAll**(payment) ⇒ Units
- makeEmptyPurse() ⇒ Purse

Purse

- withdraw(units) ⇒ Payment
- **depositAll**(payment) ⇒ Units
- getBalance() ⇒ Units
- getAssay() ⇒ Assay

```
// carol confirms ticket and pays for it
receiveTicket(ticketP) {
  const tkt = venueA.claimAll(ticketP);
  ...
  return funPurse.withdraw(moola(100));
}
```

```
// get paid by carol for a new ticket
const t = ticketsM.mint(seat("E6"));
const paymentP = carol.receiveTicket(t);
businessPurse.depositAll(paymentP);
```

ERTP Tutorial

https://agoric.com/Documentation/ertp/guide/#a-quick-tutorial

```
import { makeMint } from '@agoric/ertp/core/mint';
const baytownBucksMint = makeMint('BaytownBucks');
const purse = baytownBucksMint.mint(1000, 'community treasury');
const paymentForAlice = purse.withdraw(10, `alice's community
money`);
alice.receivePayment(paymentForAlice);
```

What Assays and Units do

- Assays – exclusive transfer
  - `claimAll` – take something you are **suspicious** of
  - On success, give back something you can **rely on**
  - Including **exclusive access** to the described assets

- Units – describe the structure of kind of asset
  - Fungible, non-fungible, set-like, etc.

Zoe:
Protecting offers since 2020

AGORIC

Zoe: Offers Everywhere!

- Seller auctions a concert ticket, a digital good
  - **Offers** a concert ticket for at least X moola
- Clients bid
  - **Offer** at most Y moola for concert ticket
- Zoe holds the offered assets
  - Async **escrow** in offer pool for contract
- Contract executes; it **reallocates** assets among offers
  - Must satisfy **offer constraints** and **conservation**
- Zoe provides **payouts**
  - Winnings and refunds

Offer Safety ensures...

... to clients

- Clients receive **desired payout** or **refund**
  ... regardless of the behavior of the contract

... to contract

- Offered assets are synchronously available
- Offered assets match their descriptions
- Assets will be delivered upon completion

Exit Safety: an offer you *can* refuse

Exit rule: what terminates the offer?

- On Demand
  - The contract code cannot obstruct exit requests
  - Even if the contract goes into an infinite lo...
- At Deadline
  - The offer is cancelled automatically at the deadline
- Up to contract
  - The client relinquishes their right to cancel

- ... or on failure
  - Contract software fault triggers exit [future]

AGORIC

Zoe offer process details

- Construct offer
  - Payout rules – assets *offered* and *wanted*
  - Exit rule – what terminates the offer?
  - Payment with offered assets
  - Returns escrow receipt and payout promise
- Pass escrow receipt to contract as the offer
- On completion, payout promise resolves to:
  - On success: **winnings** as specified in *wanted*
  - On rejection: **refund** of *offered* assets

Zoe framework operations

For clients

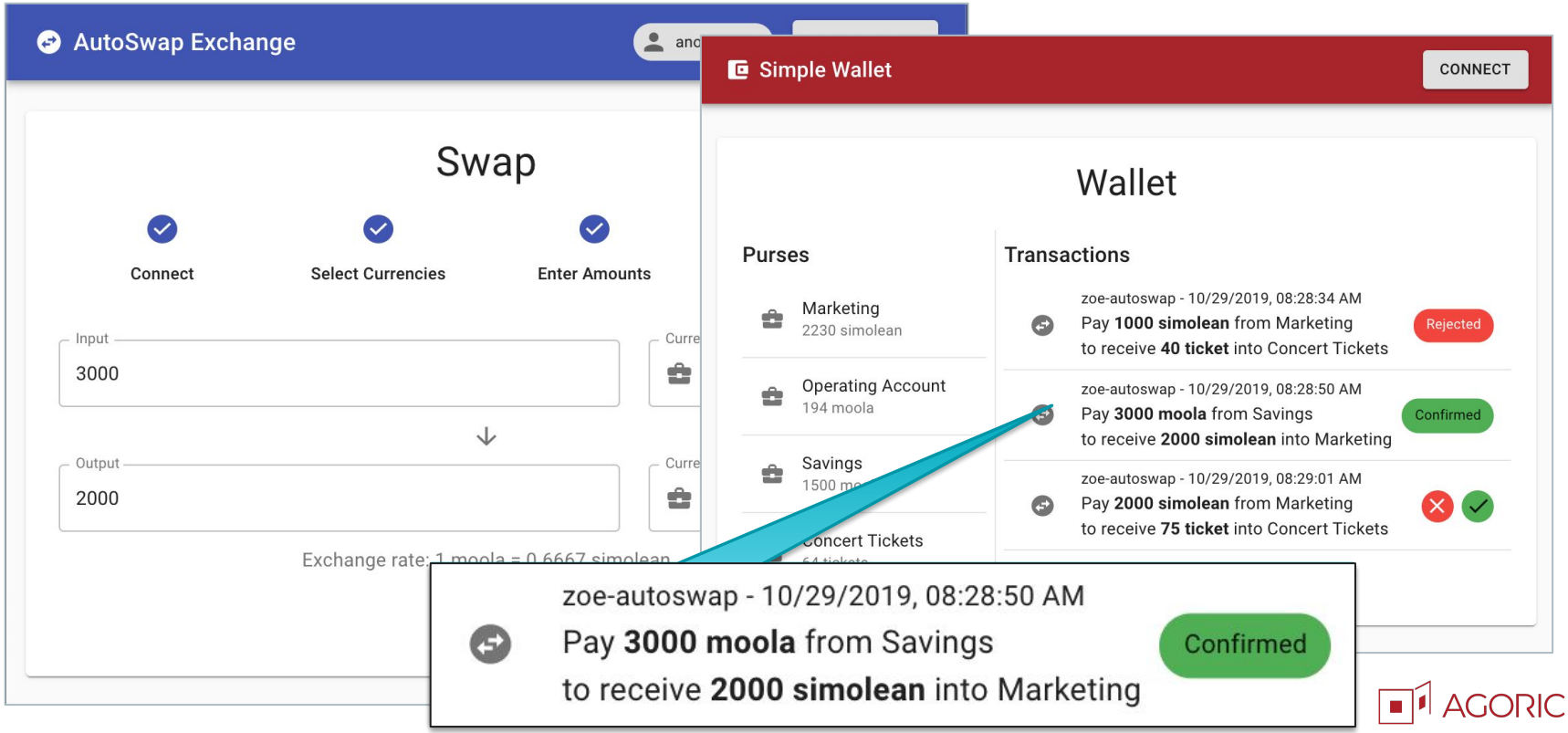
- Trusted path to contracts
- Escrow offers

  (assets and offer terms)
- Ensure exit
- Provide payouts

For contracts

- Instantiate contract
- React to available offers
- Reallocate assets
  - Consistent with offer terms
  - Ensures conservation

Offer-based wallets



AGORIC

Buy a ticket with an offer

```
// Lowest level version (few helpers)

// setup for concert tickets
const wanted = seat(‘E4’);
const offerAtMost = moola(100);
const offerRules = {
  payoutRules: [
    { kind: 'wantExactly', units: wanted },
    { kind: 'offerAtMost', units: offerAtMost }]
};

const { receipt, payout } = zoe.escrowOffer(offerRules, myMoola.withdraw(offerAtMost));
// user code
processEventualPayout(payout);
await auction.bid(receipt);
```

Writing A Zoe Smart Contract

https://agoric.com/Documentation/zoe/guide/#what-is-zoe

```
export function makeContract(zoe, terms) {
 return {
   instance: {
     async makeOffer(escrowReceipt) {
       const { offerHandle } = await zoe.burnEscrowReceipt(escrowReceipt);
       // I have no business logic!
       zoe.complete([offerHandle]);
     },
   },
   assays: terms.assays,
 };
};
```

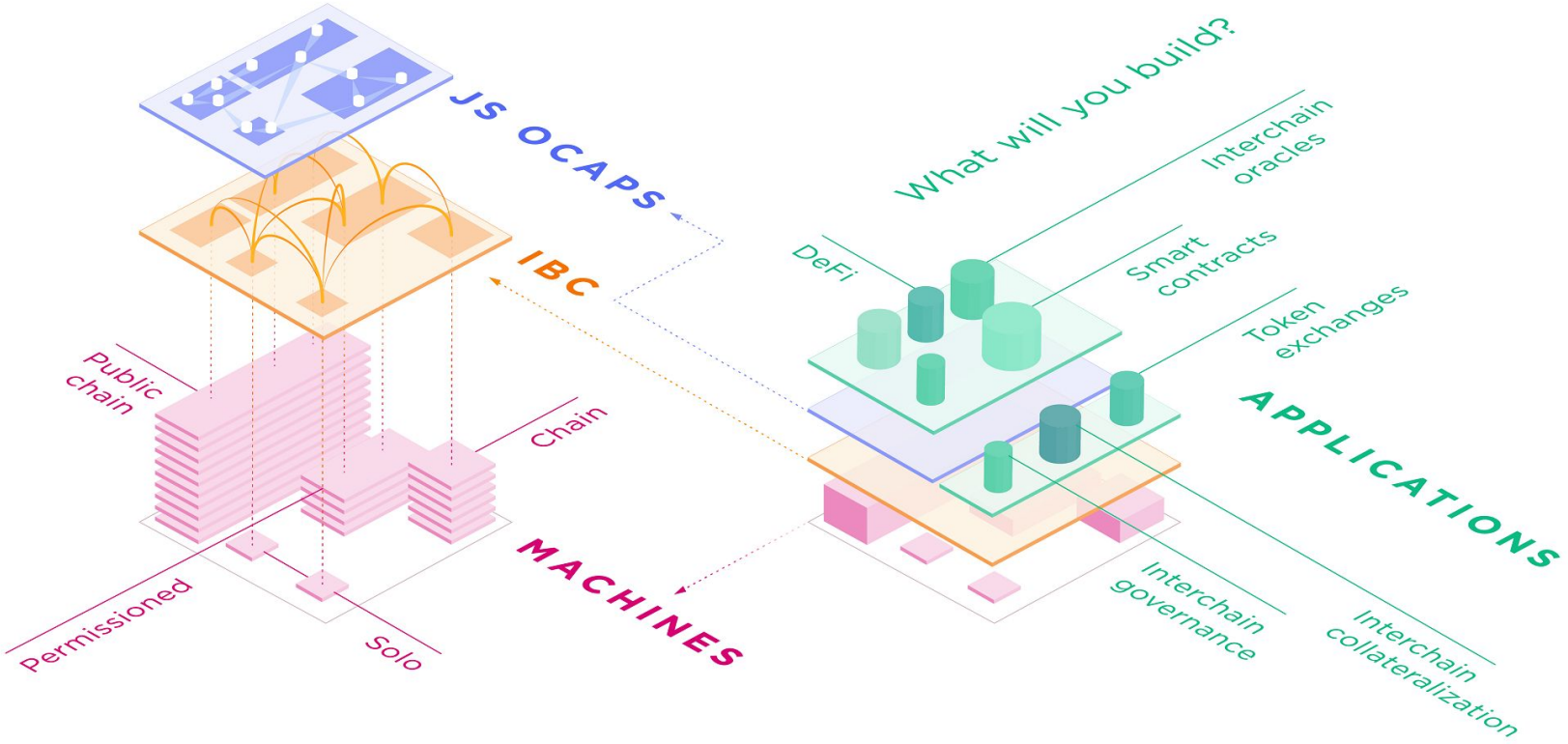
Writing a Zoe Smart Contract

https://github.com/Agoric/ERTP/blob/master/core/zoe/contracts/simpleExchange.js

Simple Exchange

Limit orders for an asset, can buy and sell.

https://agoric.com/Documentation/zoe/guide/contracts/simple-exchange.html


JS OCAPS
IBC
MACHINES
Public chain
Chain
Permissioned
Solo
What will you build?
Interchain oracles
DeFi
Smart contracts
Token exchanges
APPLICATIONS
Interchain governance
Interchain collateralization