Transaction Updates

Input: timestamp,
Balance: 500, *signature*,
sender's public key: 0xfoo1

Output:
Amount: 40,
Address: 0xbar2

Output:
Amount: 440,
Address: 0xfoo1

Transaction Updates

Send 20 to 0xfoo3

Input: timestamp,
Balance: 500, *signature*,
sender's public key: 0xfoo1

Output:
Amount: 40,
Address: 0xbar2

Output:
Amount: 440,
Address: 0xfoo1

Input: timestamp,
Balance: 500, *signature*,
sender's public key: 0xfoo1

Output:
Amount: 20,
Address: 0xfoo3

Output:
Amount: 420,
Address: 0xfoo1

Transaction Updates

Send 20 to 0xfoo3

<u>Output</u>:
Amount: 40,
Address: 0xbar2

<u>Input</u>: timestamp,
Balance: 500, *signature*,
sender's public key: 0xfoo1

<u>Output</u>:
Amount: 20,
Address: 0xfoo3

<u>Output</u>:
Amount: 420,
Address: 0xfoo1

Consider the case where an individual wants to create another transaction in a short timeframe.

We could easily create an entirely new transaction object to represent the new exchange but in this case the input would virtually be the same between the original and new transaction.

So rather than creating an entirely new transaction object with a whole new ID and other it can optimize the process.

Instead we could update the original transaction with the new output to represent the new exchange.

Meaning there is one transaction object that consent currency to more than one individual to multiple outputs with an additional output.

He will resign the transaction to update the input of the new signature.

Likewise the output will subtract the sum total of all the outputs.

They're sending it out for the one that they're sending as the resulting output or at the covers a quick