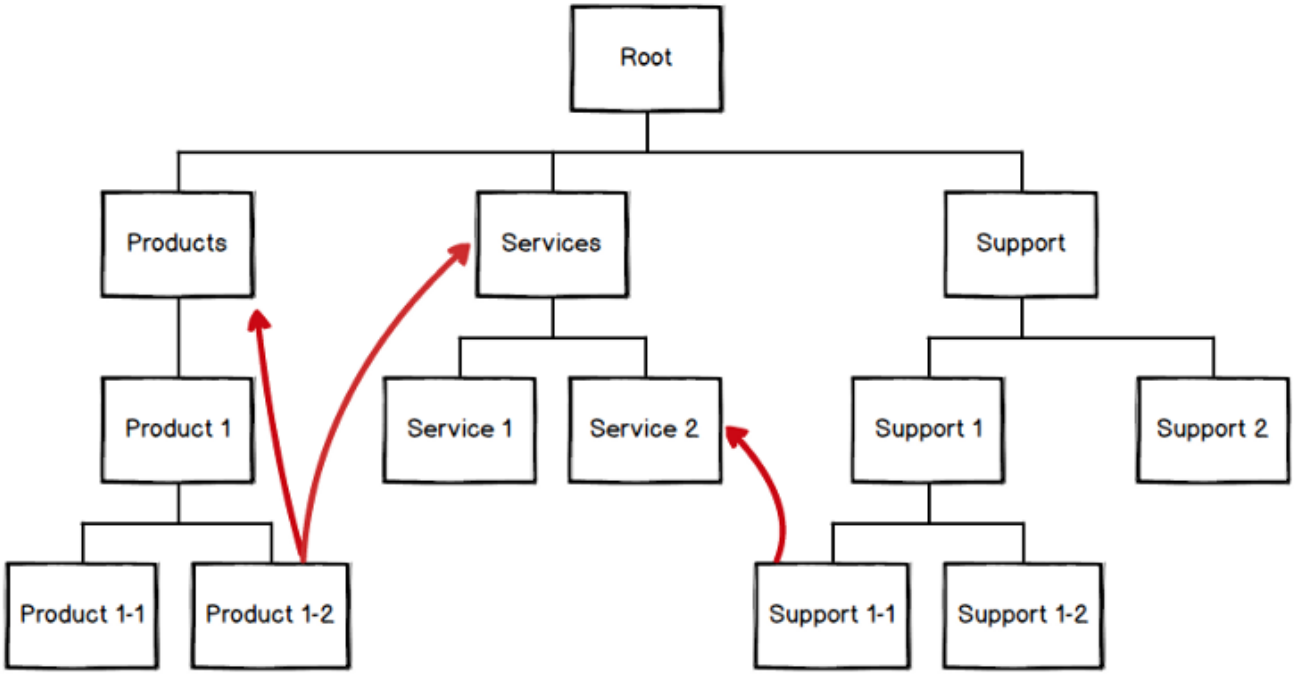
2.3 Data flow in Redux

As you saw in the part 1, in React, the data is passed through the component using props. This is called **unidirectional data flow** that flows from parent to child.

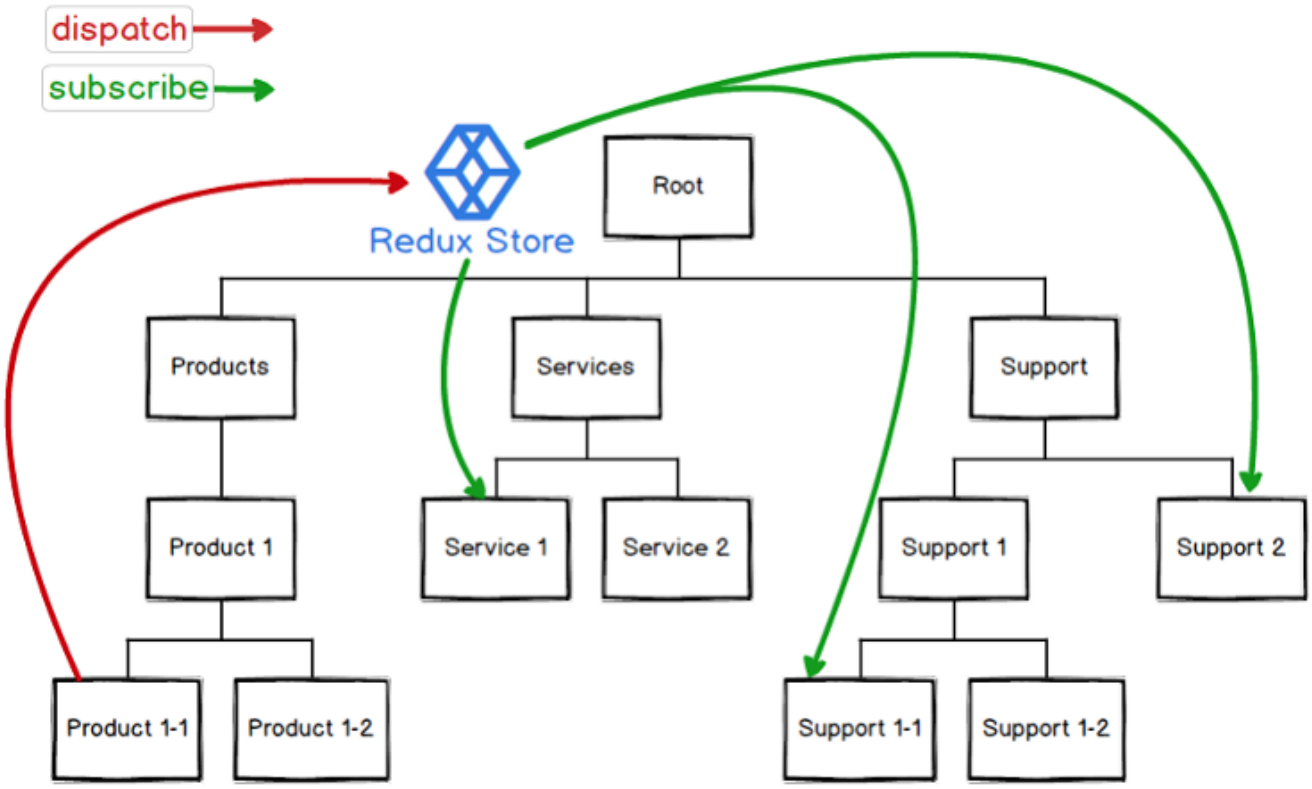
Due to these characteristics, communication between components other than parent-child relationship is not clear.



React does not recommend direct **component-to-component**communication as shown above. There is a suggested way for this in React, but you have to implement it yourself.

According to **React docs**:

> *For communication between two components that don't have a parent-child relationship, you can set up your own global event system. ... Flux pattern is one of the possible ways to arrange this.*



This is where Redux comes in handy.

Redux provides a solution for managing all application state in a single place called a `store`.

The component then `dispatches` the state change to the store instead of passing it directly to the other components

The components that need to be aware of state changes can `subscribe` to the store.

> *Redux is, in a word, a **state container** that represents and manages the state of an app as a single object from a JavaScript-based application.*