React Advanced

Содержание

Содержание

Что мы ждем от интерфейсов?

Что мы ждем от интерфейсов?

1. **Помогают решать наши проблемы**

Что мы ждем от интерфейсов?

1. Помогают решать наши проблемы
2. **Комфортны в использовании**

Что мы ждем от интерфейсов?

1. Помогают решать наши проблемы
2. **Не лагают, не дергаются и тд**

А почему некоторые интерфейсы лагают?

Кто-то написал плохой код

Кто-то написал плохой код)
Возможно)

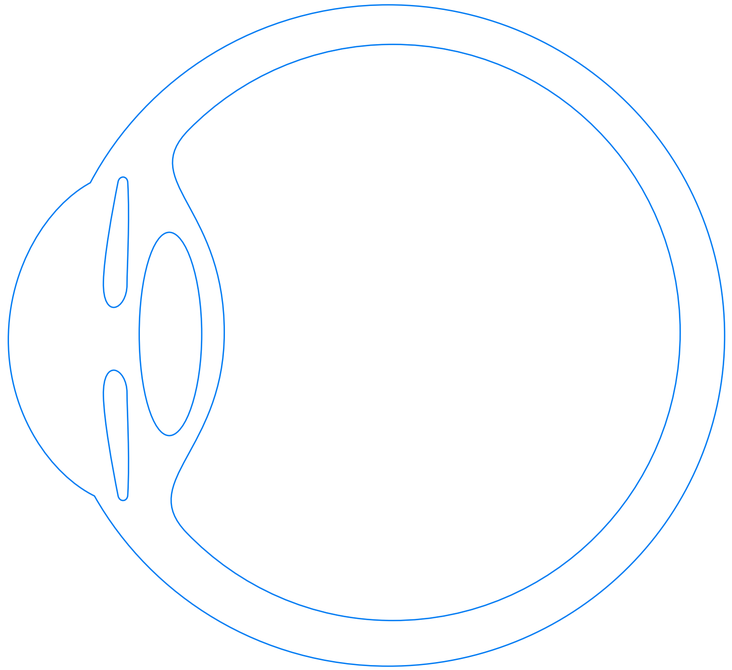
Кадры меняются очень медленно

Кадры меняются медленнее чем нужно

А сколько нужно?

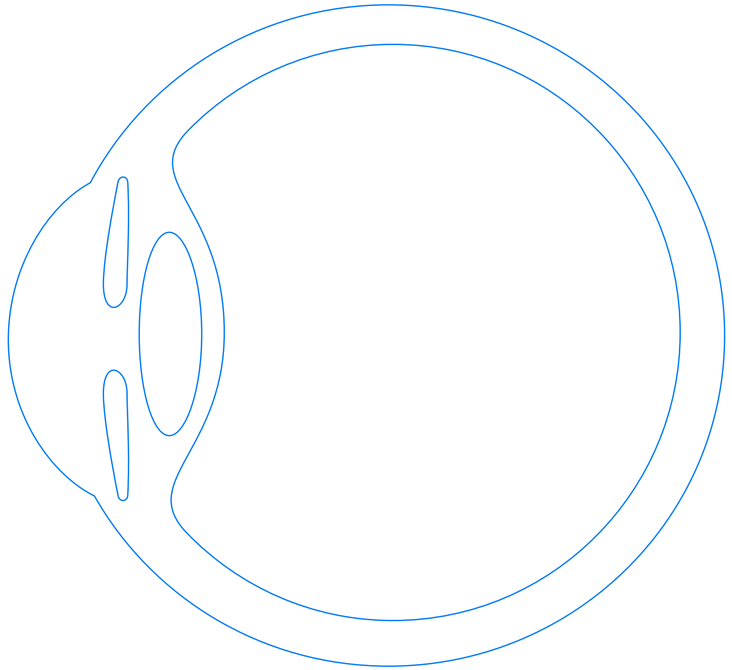
Зависит от того, кто ваш пользователь

Насекомые - до 250ГЦ



Насекомые - до 250ГЦ

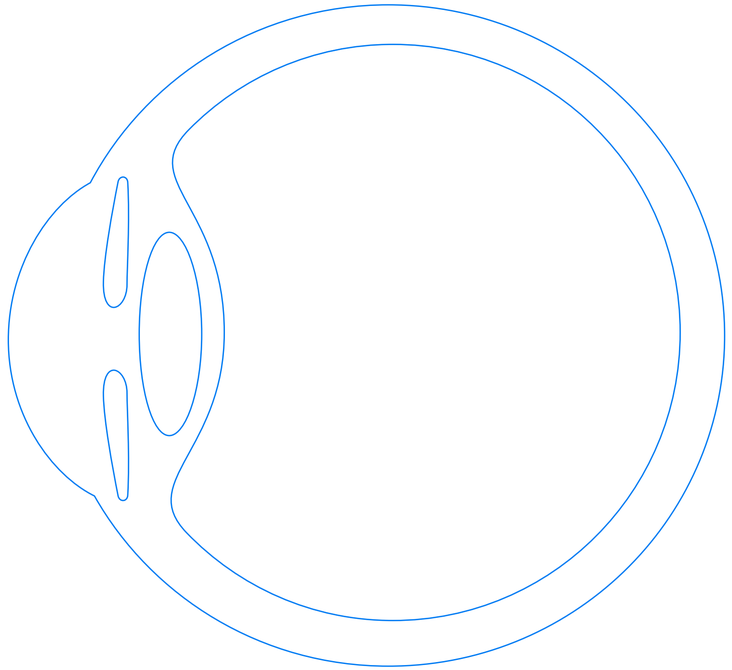
Суслики - около 120ГЦ



Насекомые - до 250ГЦ

Суслики - около 120ГЦ

Собаки и птицы - около 80ГЦ

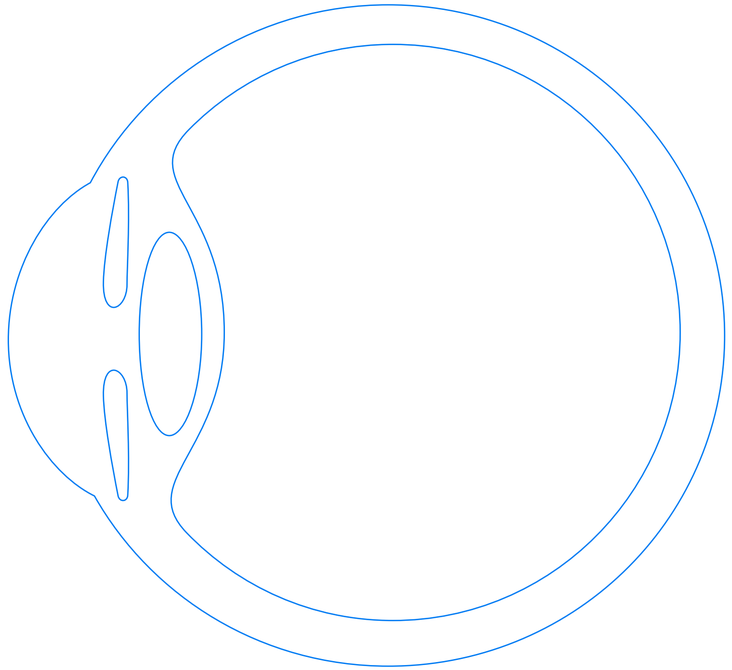


Насекомые - до 250ГЦ

Суслики - около 120ГЦ

Собаки и птицы - около 80ГЦ

Человек - около 60ГЦ



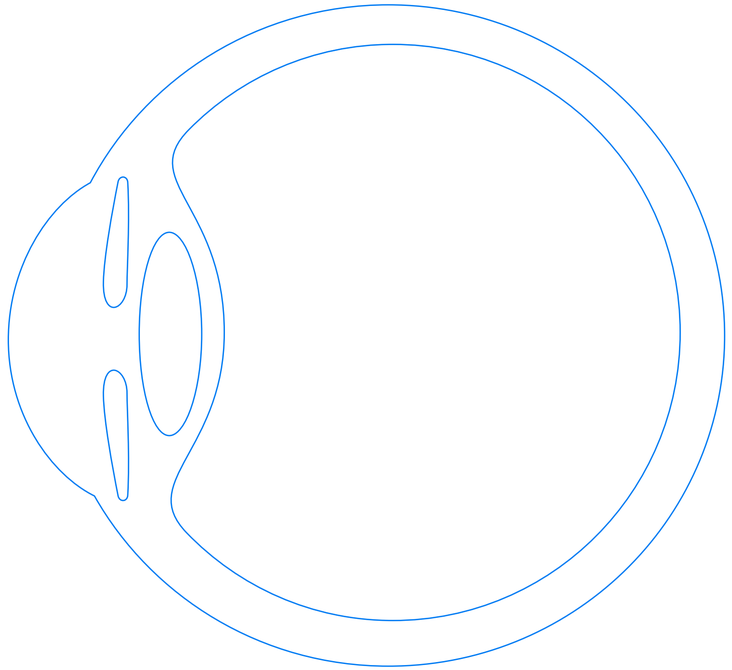
Насекомые - до 250ГЦ

Суслики - около 120ГЦ

Собаки и птицы - около 80ГЦ

Человек - около 60ГЦ

Угри - около 14ГЦ



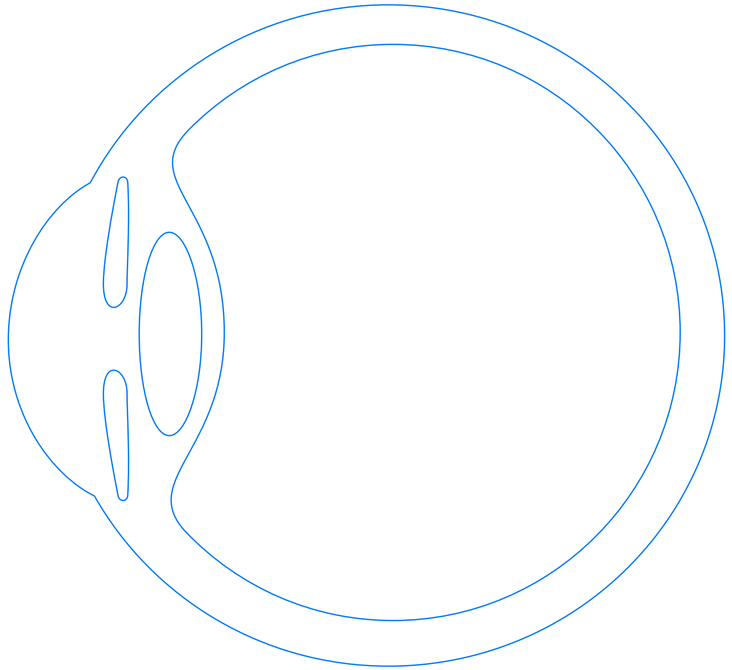
Насекомые - до 250ГЦ

Суслики - около 120ГЦ

Собаки и птицы - около 80ГЦ

Человек - около 60ГЦ

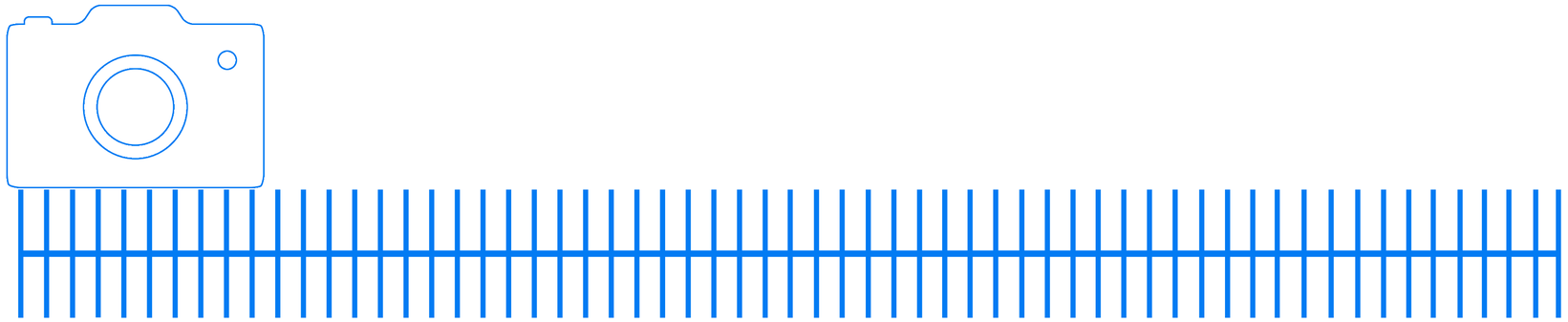
Угри - около 14ГЦ



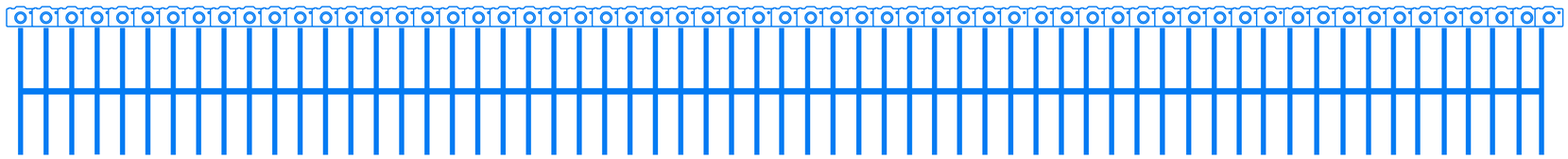
60ГЦ

60 - кадров в секунду

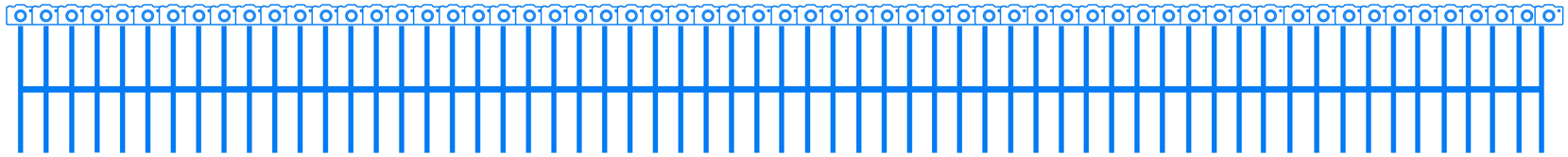
60 - кадров в секунду



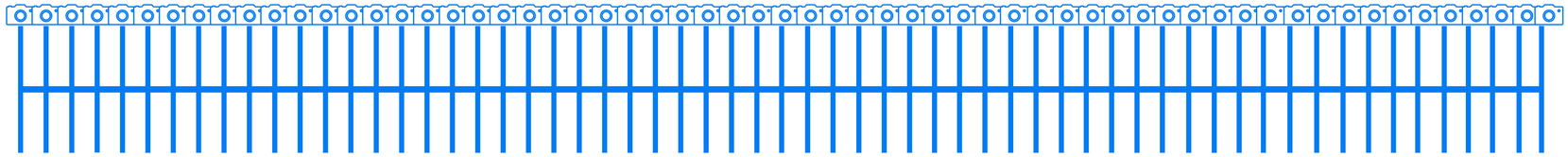
60 - кадров в секунду



А кто этим будет заниматься?



React, конечно)



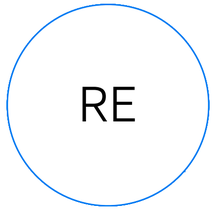
Разберемся как рендерится наш интерфейс

Весь путь разделяется на 2 фазы

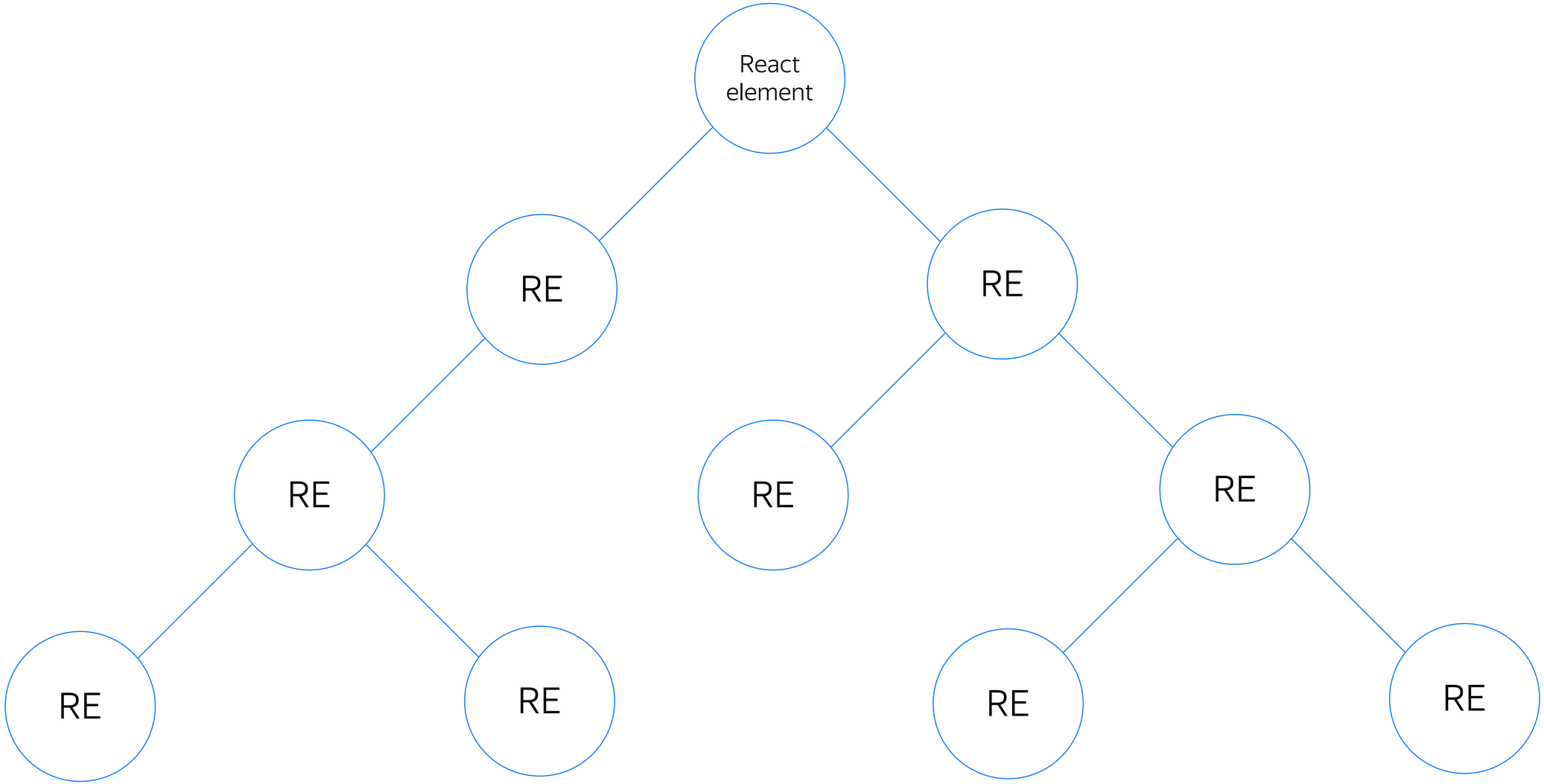
Фаза 1
Rendering and Reconciliation

Фаза 1
Рендеринг и Сравнение

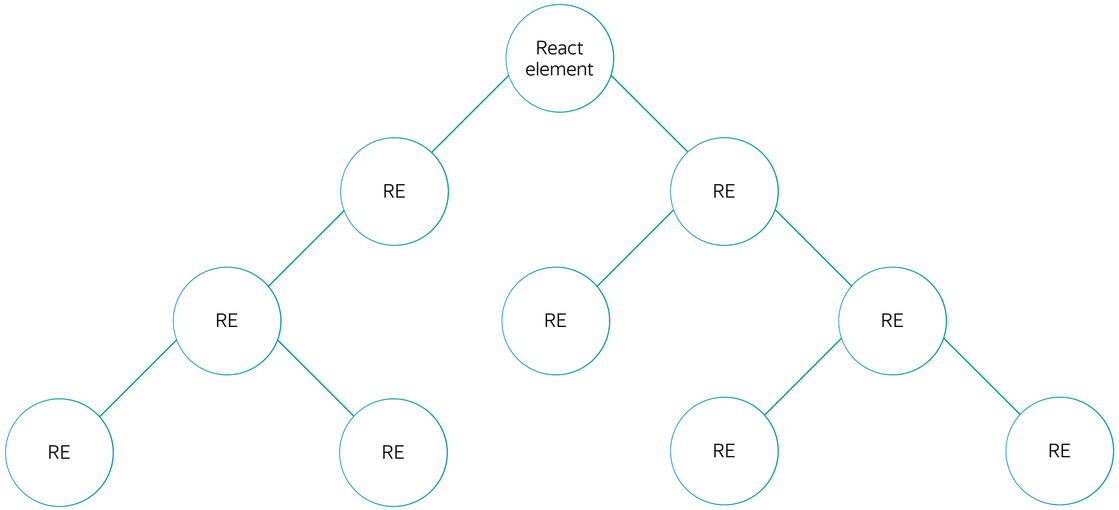
Все начинается с элементов



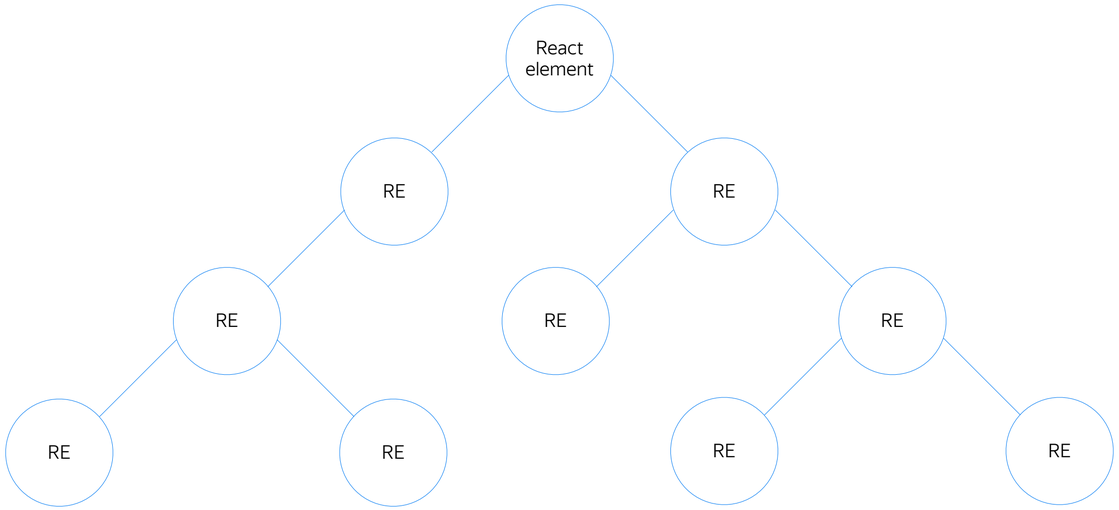
Все начинается с дерева элементов

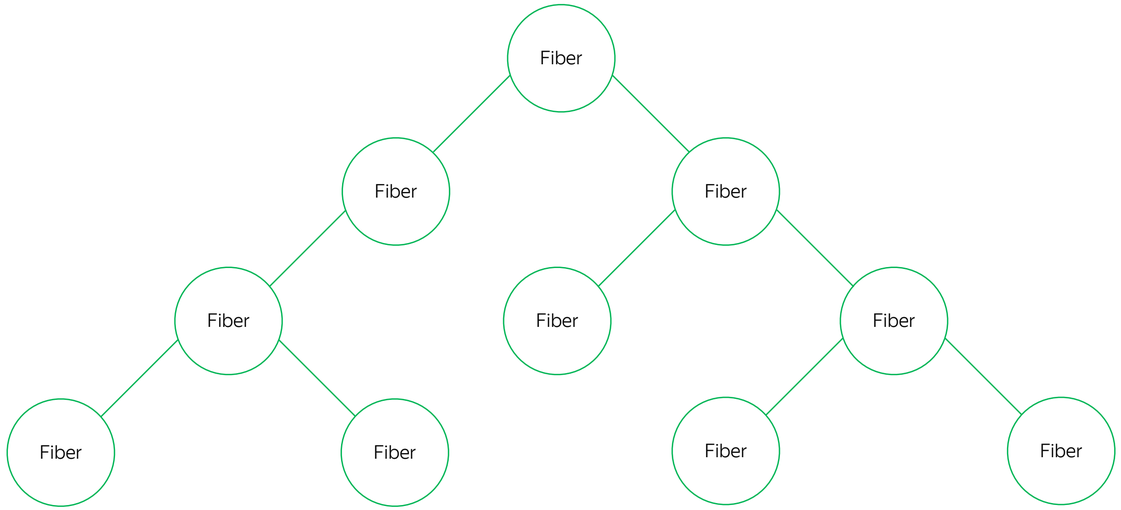


Но дерево элементов не единственное дерево!

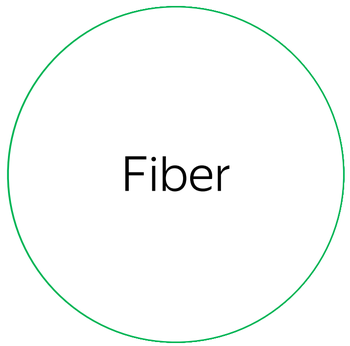


Существует дерево волокон

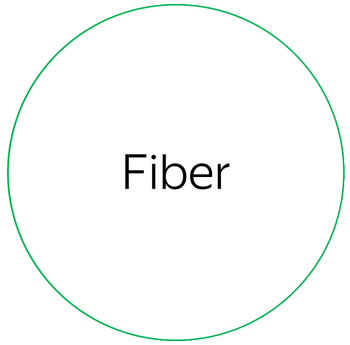




Fiber - волокно

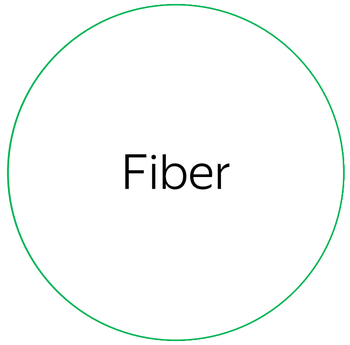


Fiber - волокно



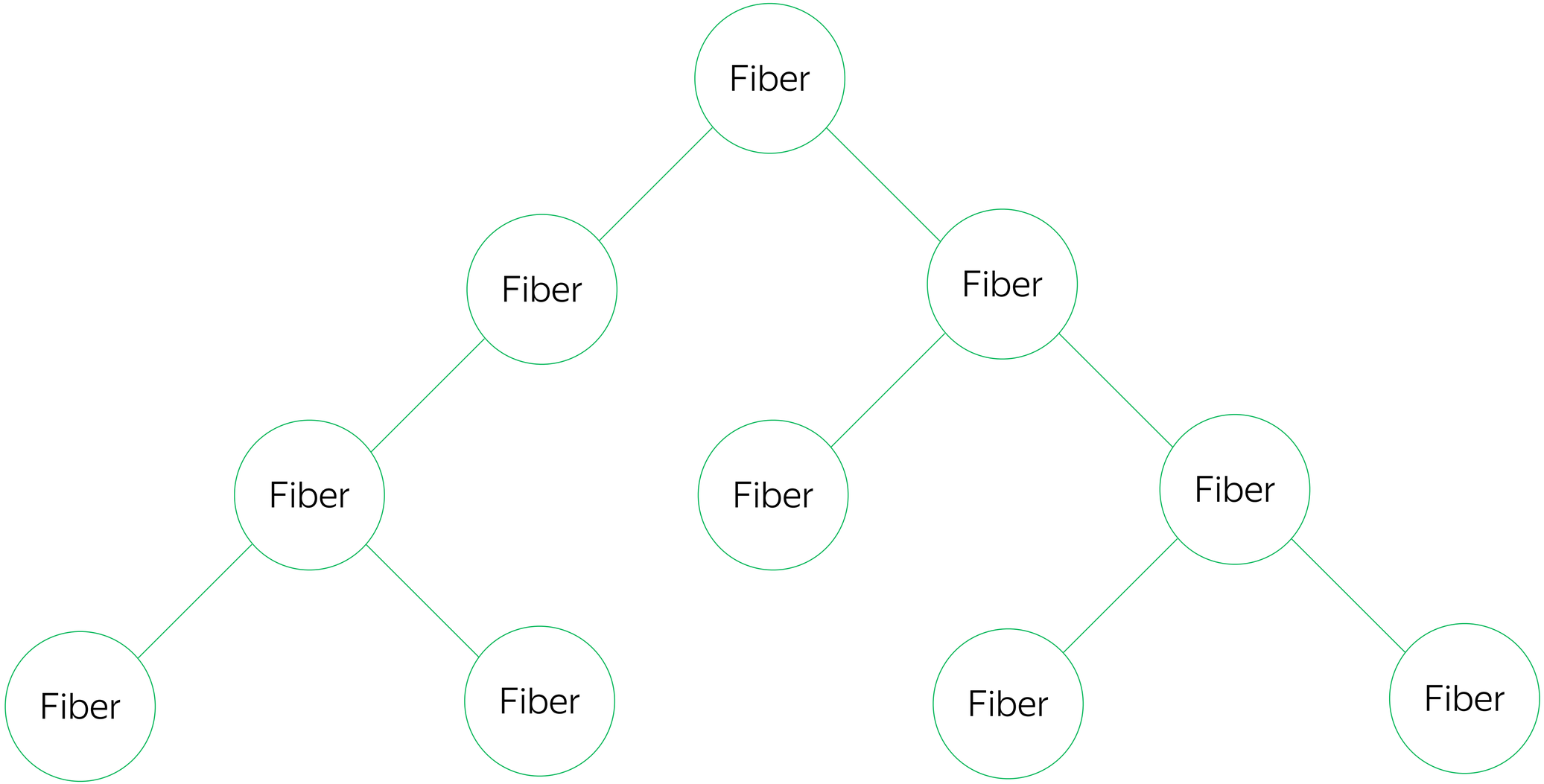

```
{
  stateNode,
}
```

Fiber - волокно

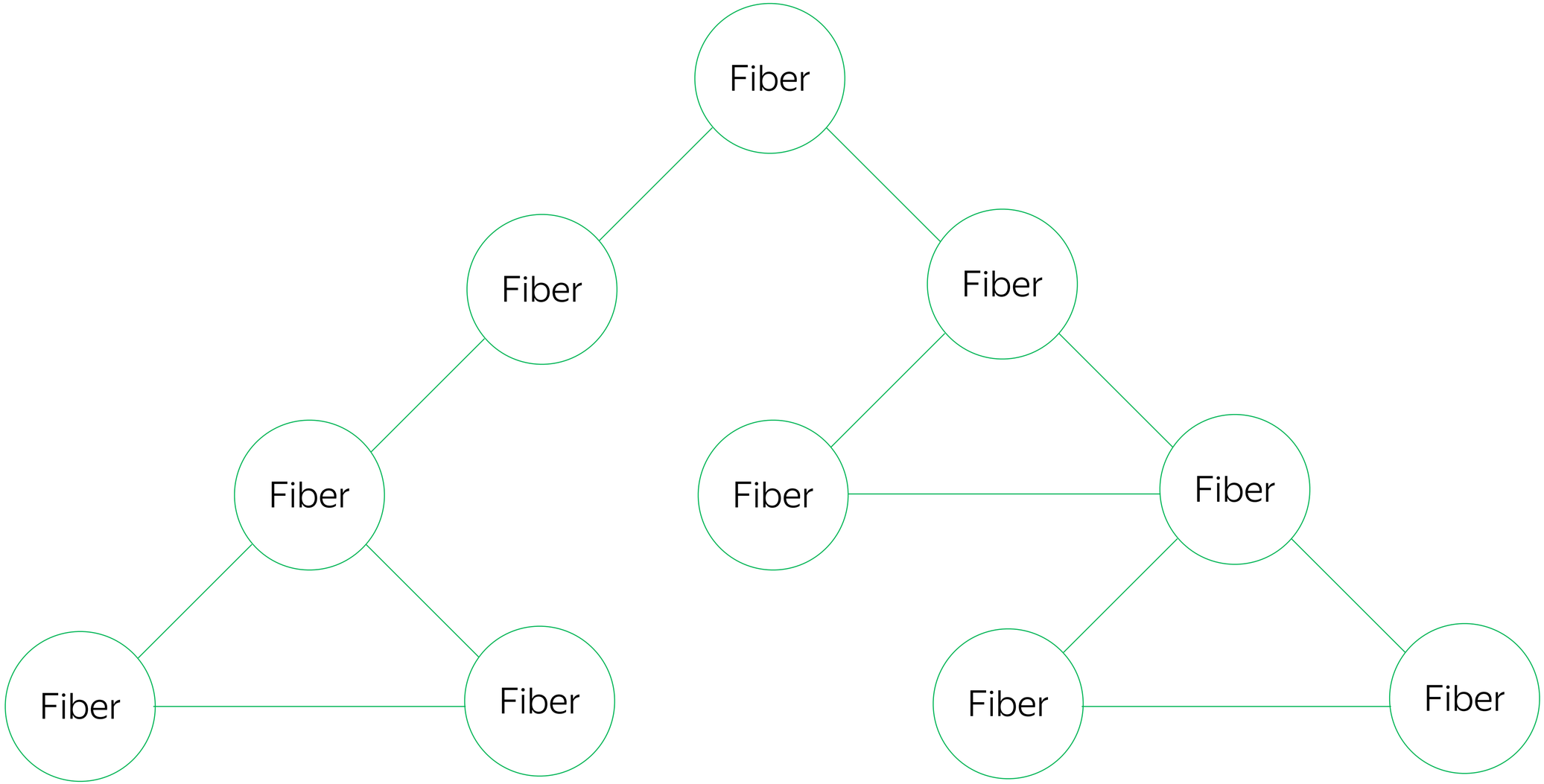



```
{
  stateNode,
  memoizedProps,
  memoizedState,
}
```

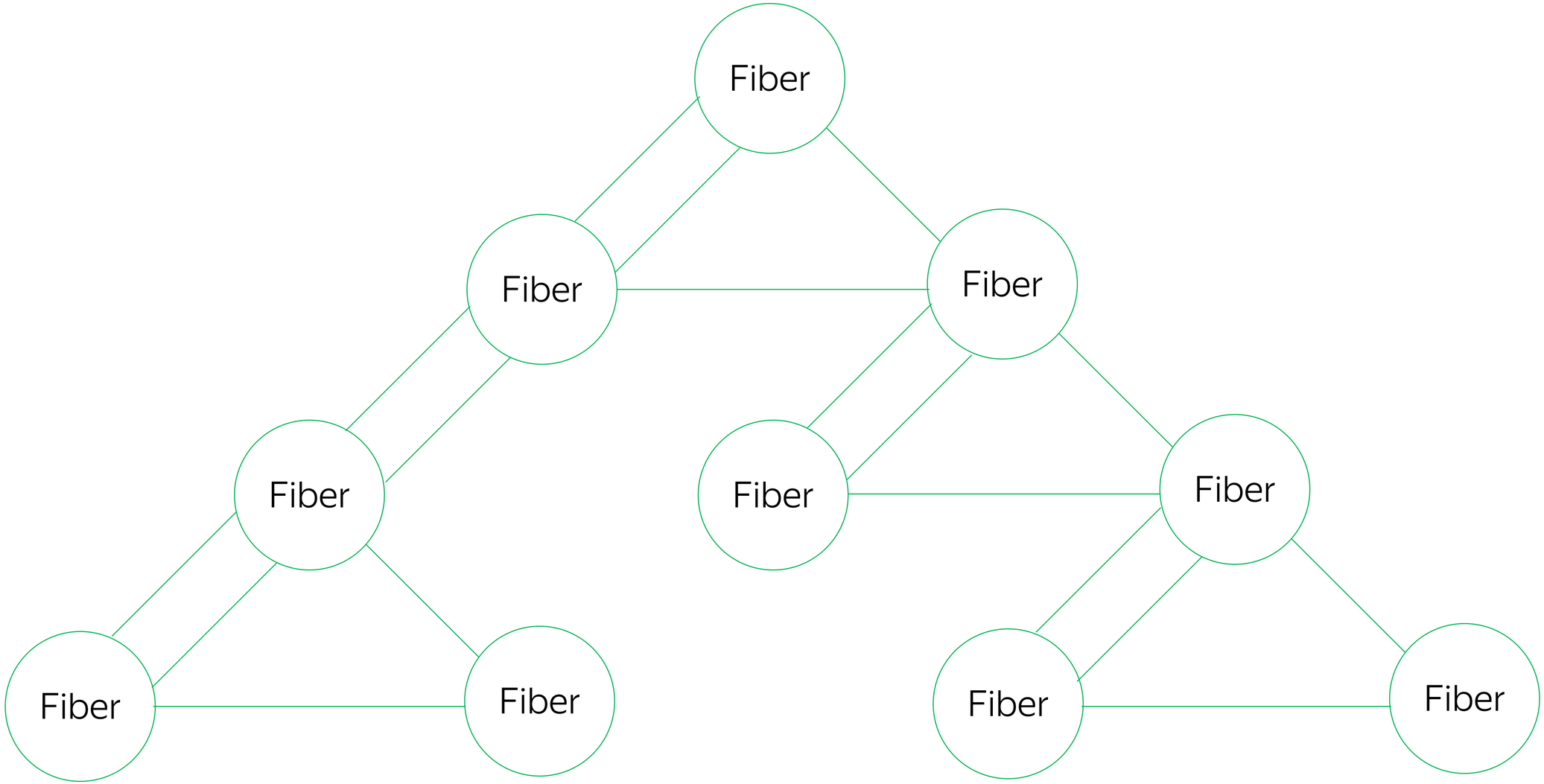
Fiber - tree?


Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

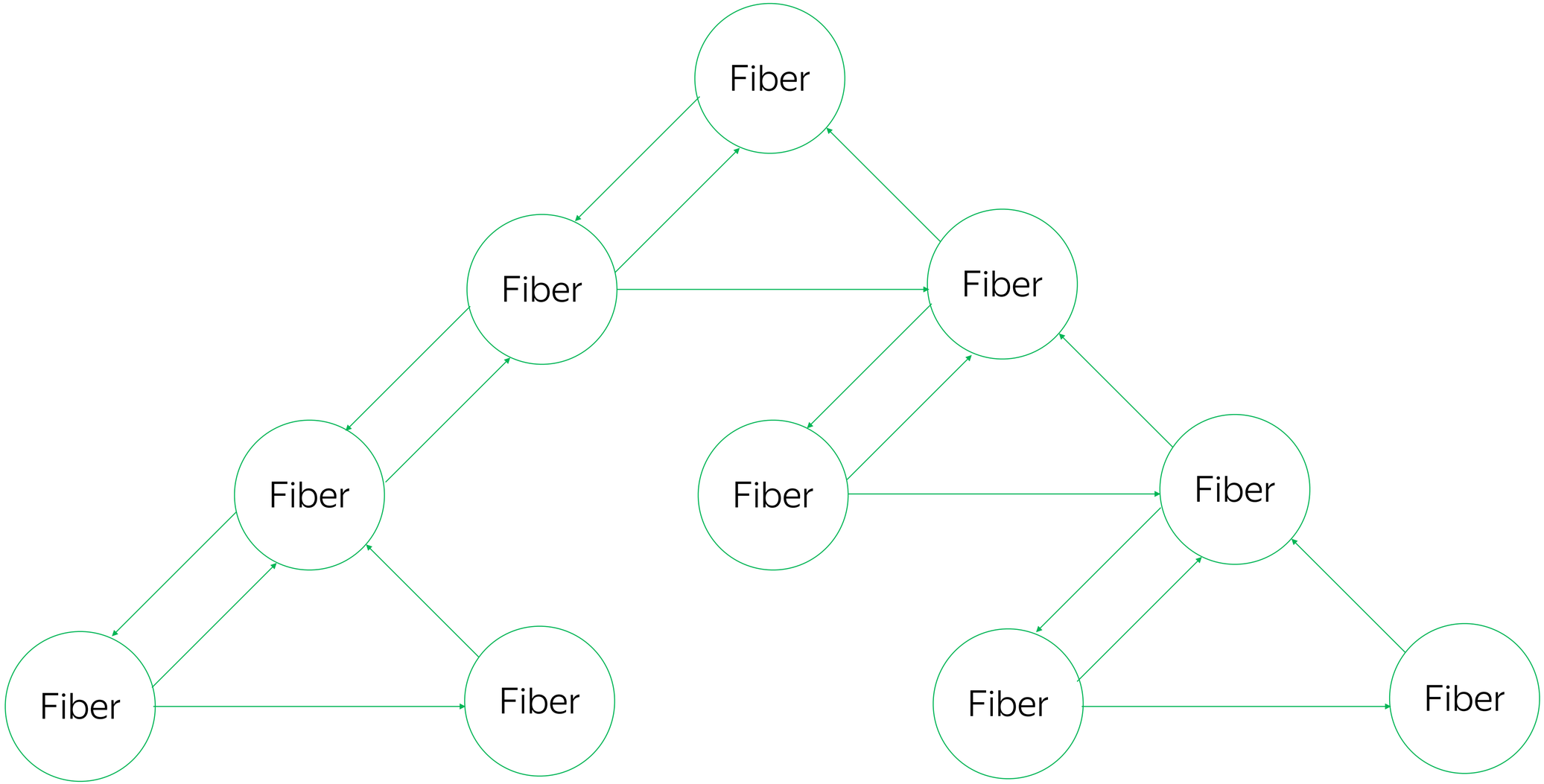
Fiber - tree?


Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

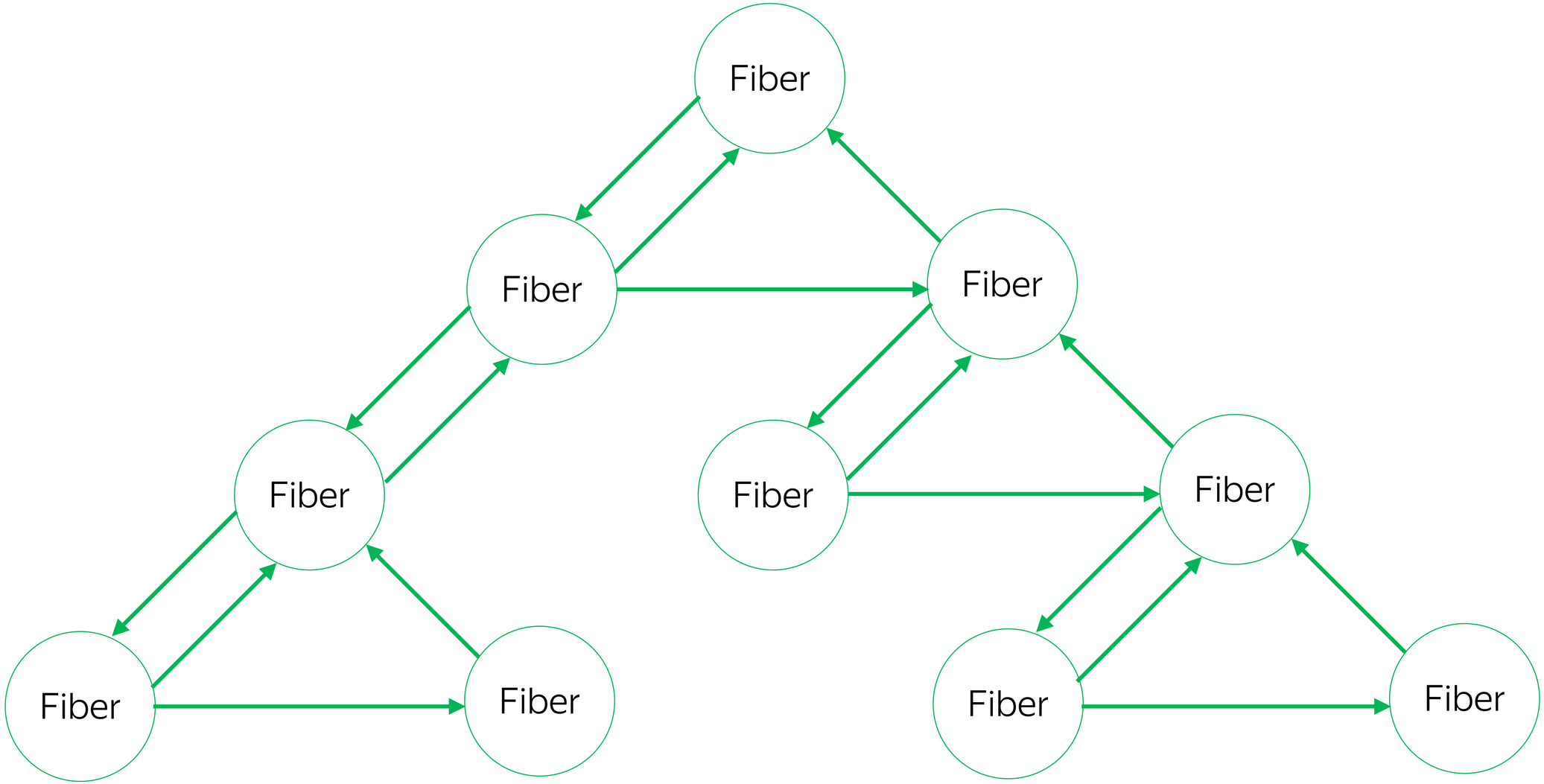
Fiber - tree?



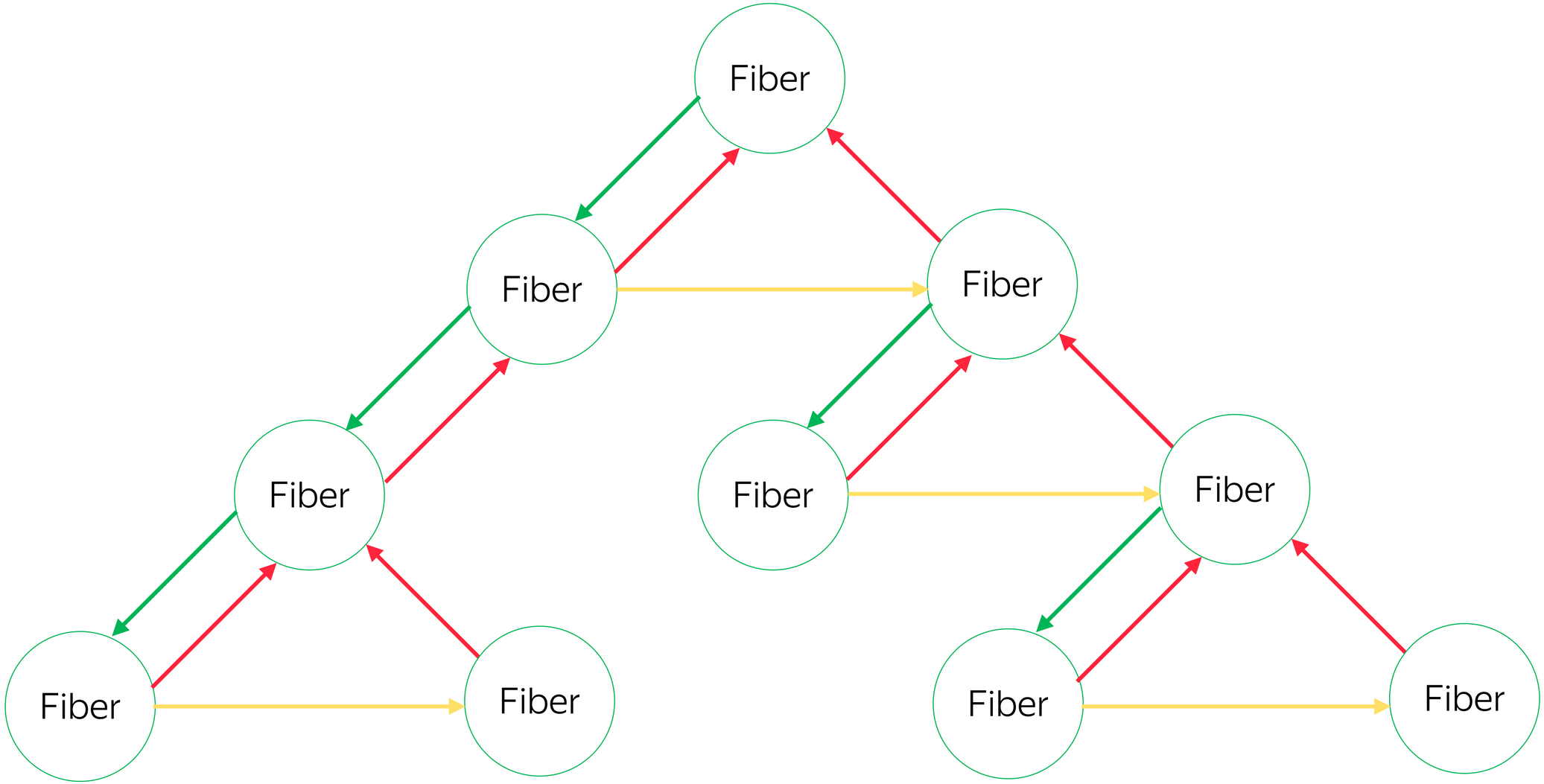
Fiber - tree?


Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

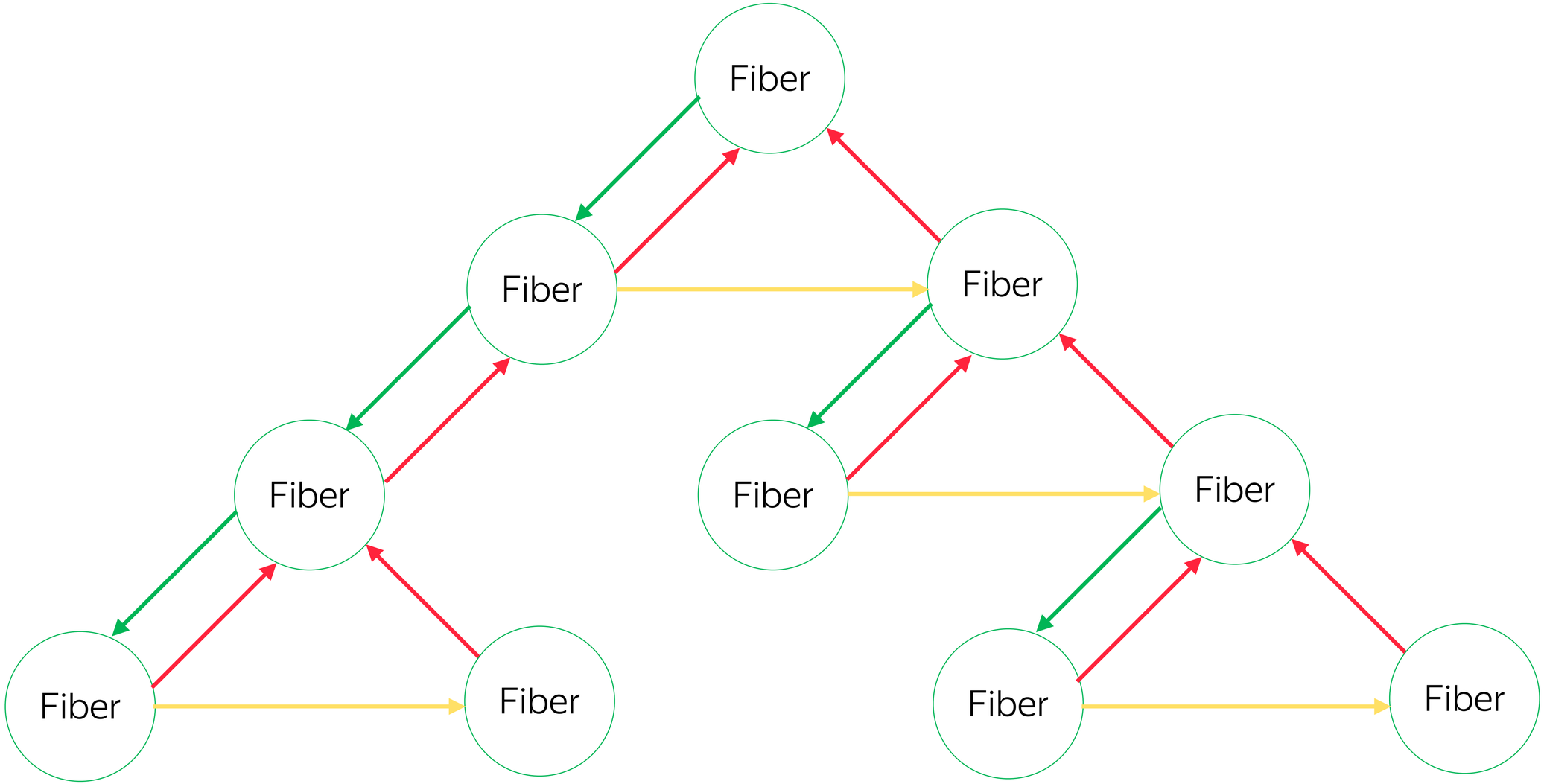
Fiber - tree?


Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

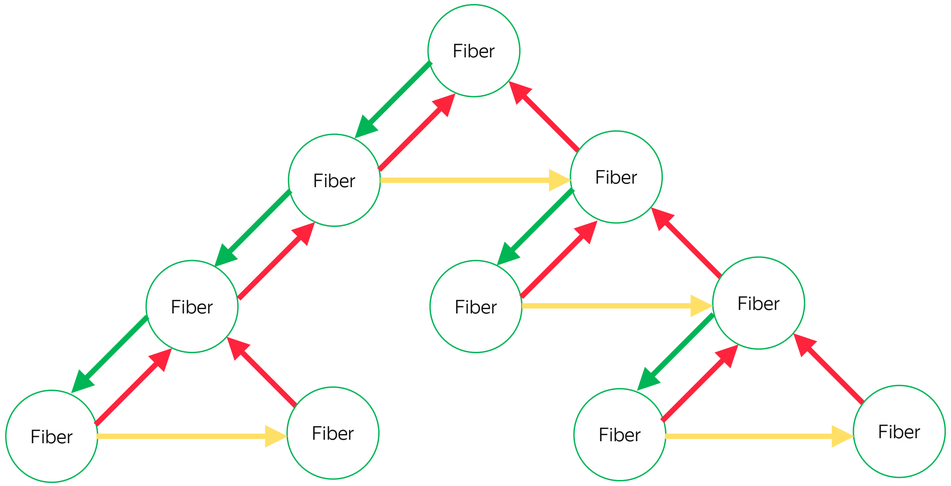
Fiber - tree?


Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

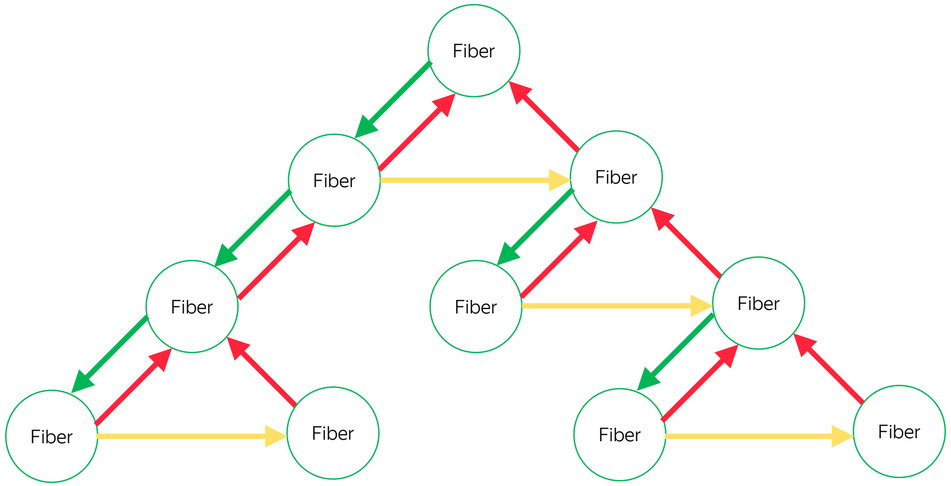
Fiber - Linked List

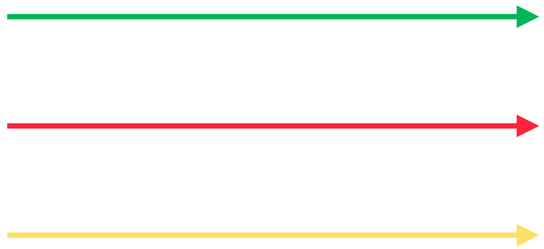


Relationships

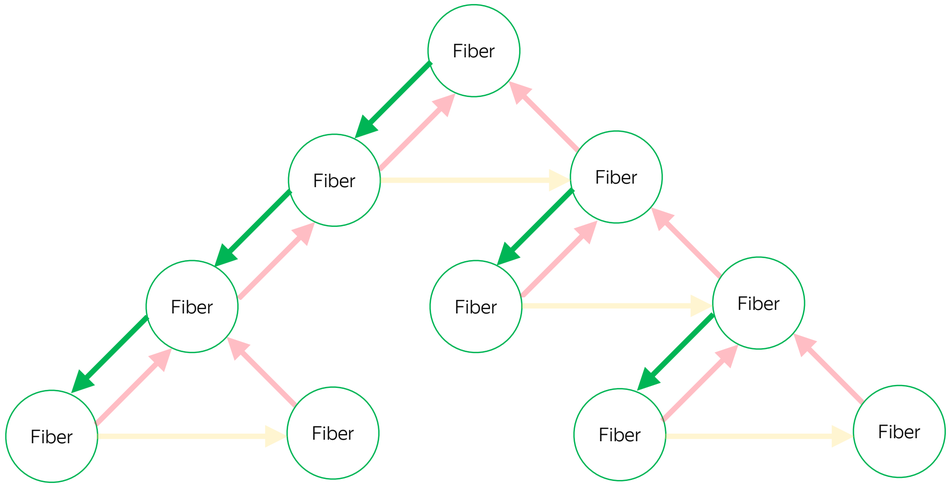


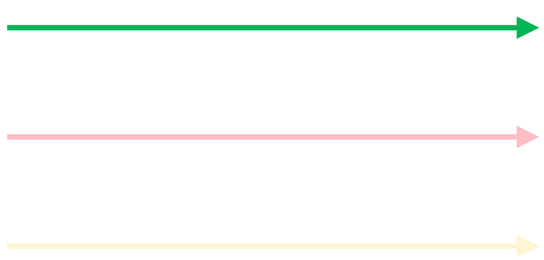
Relationships





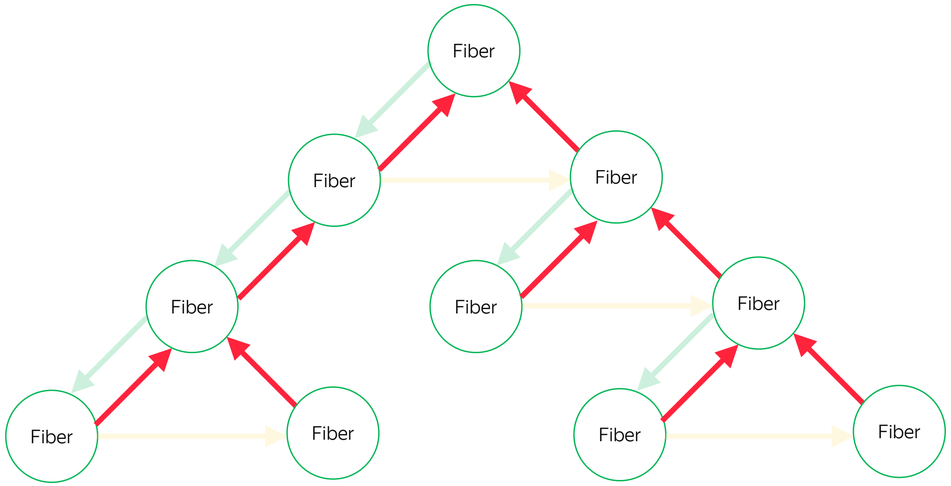
Relationships





Child

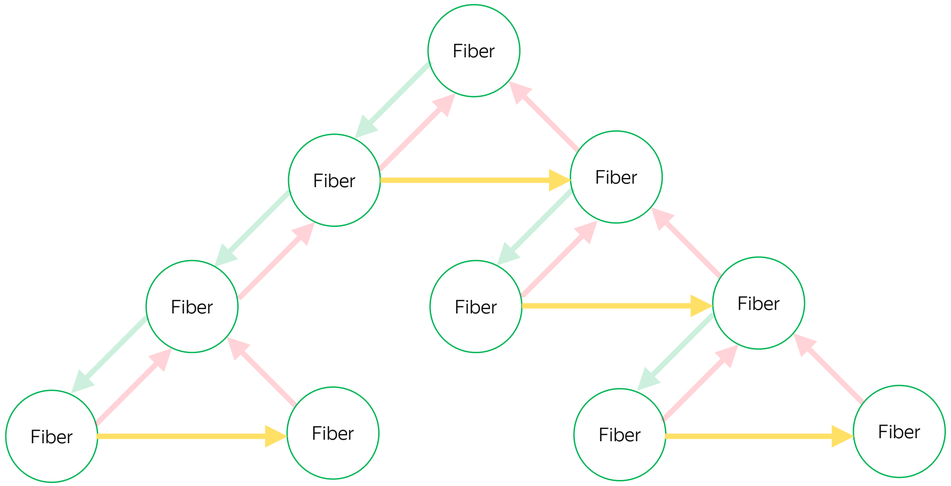
Relationships



Child

Return(parent)

Relationships

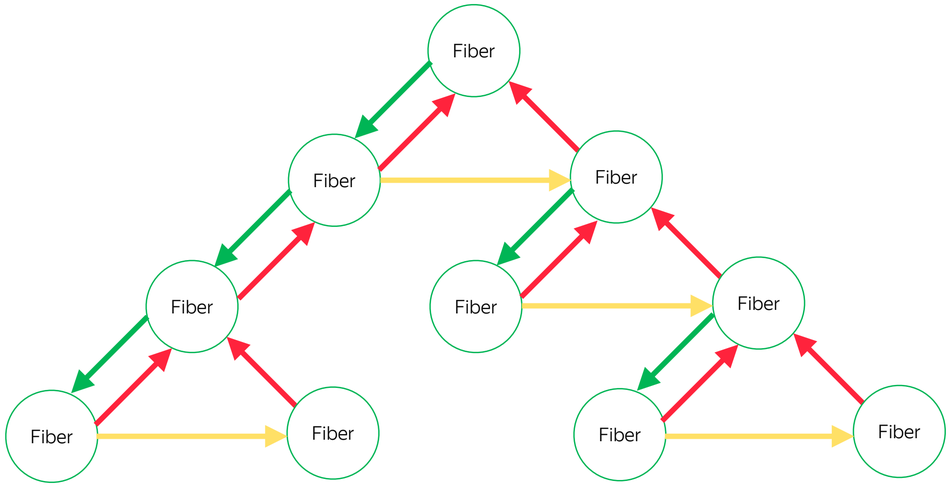


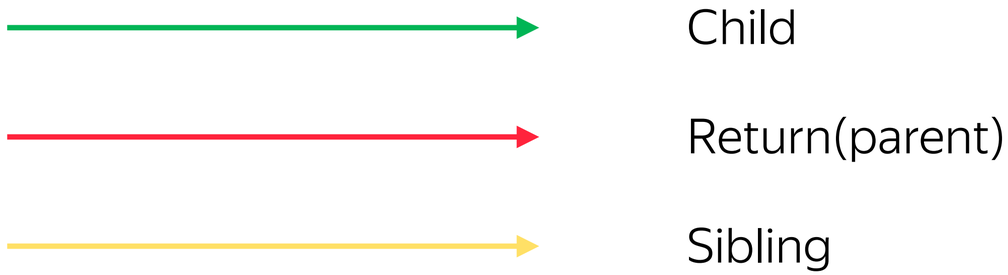
Child

Return(parent)

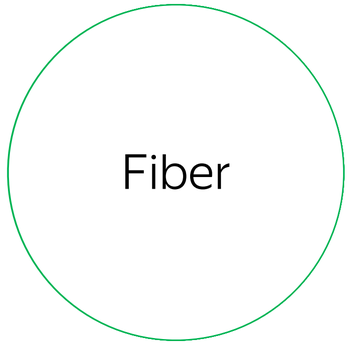
Sibling

Relationships



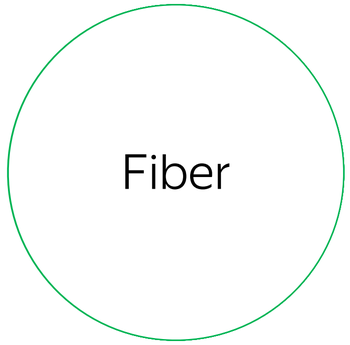


Fiber - волокно



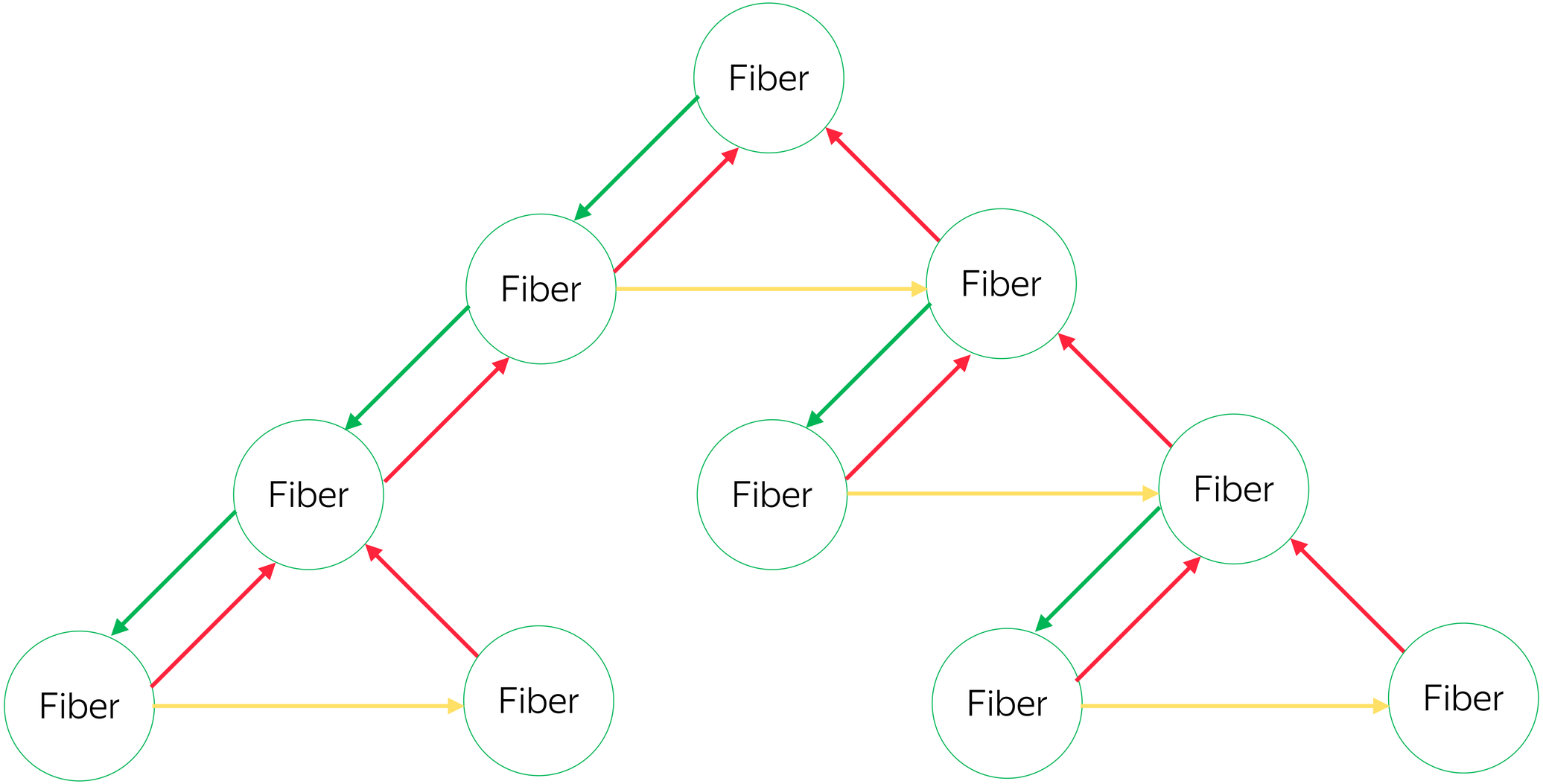

```
{
  stateNode,
  memoizedProps,
  memoizedState,
}
```

Fiber - волокно

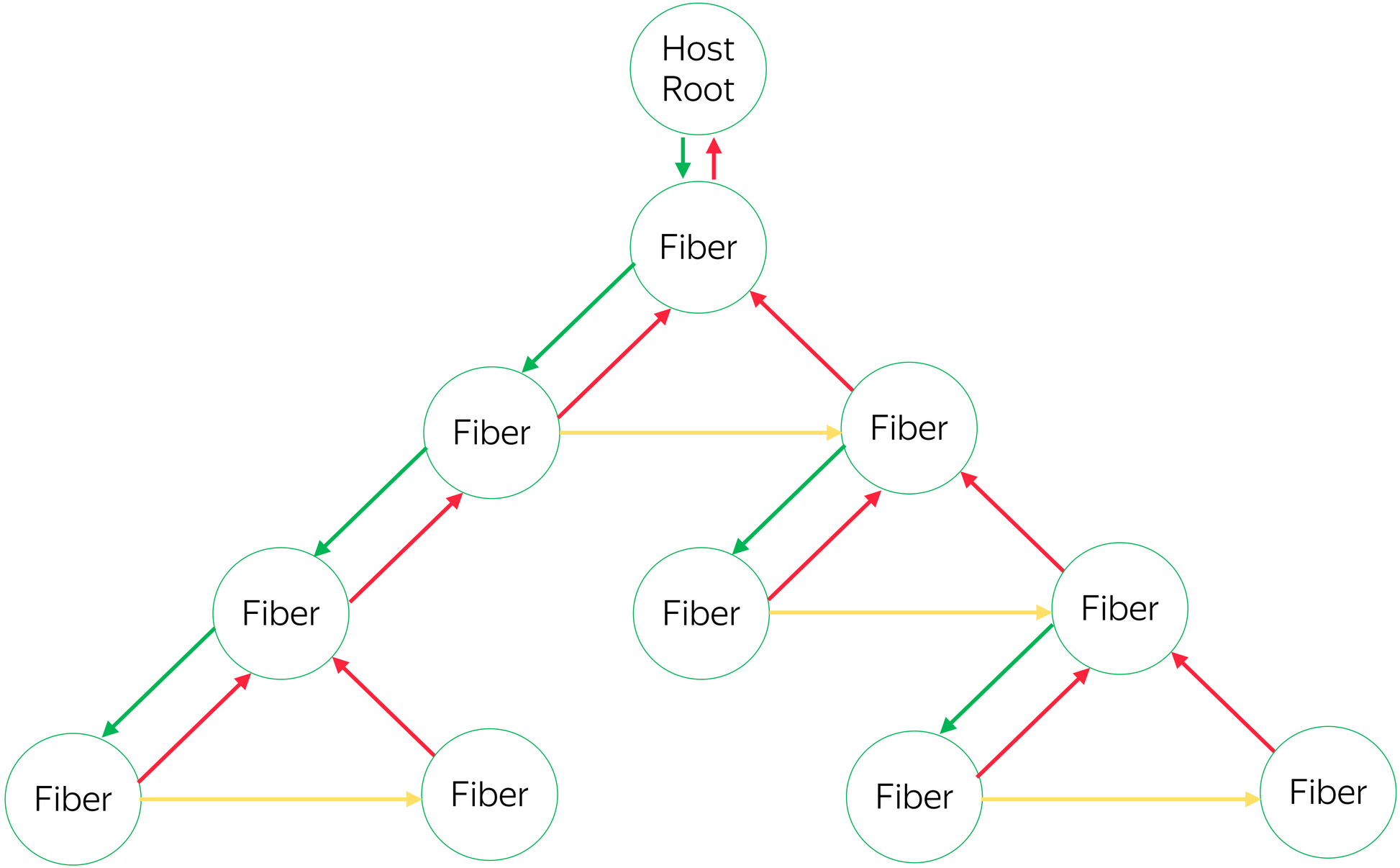



```
{
  stateNode,
  memoizedProps,
  memoizedState,
  child,
  return,
  sibling,
}
```

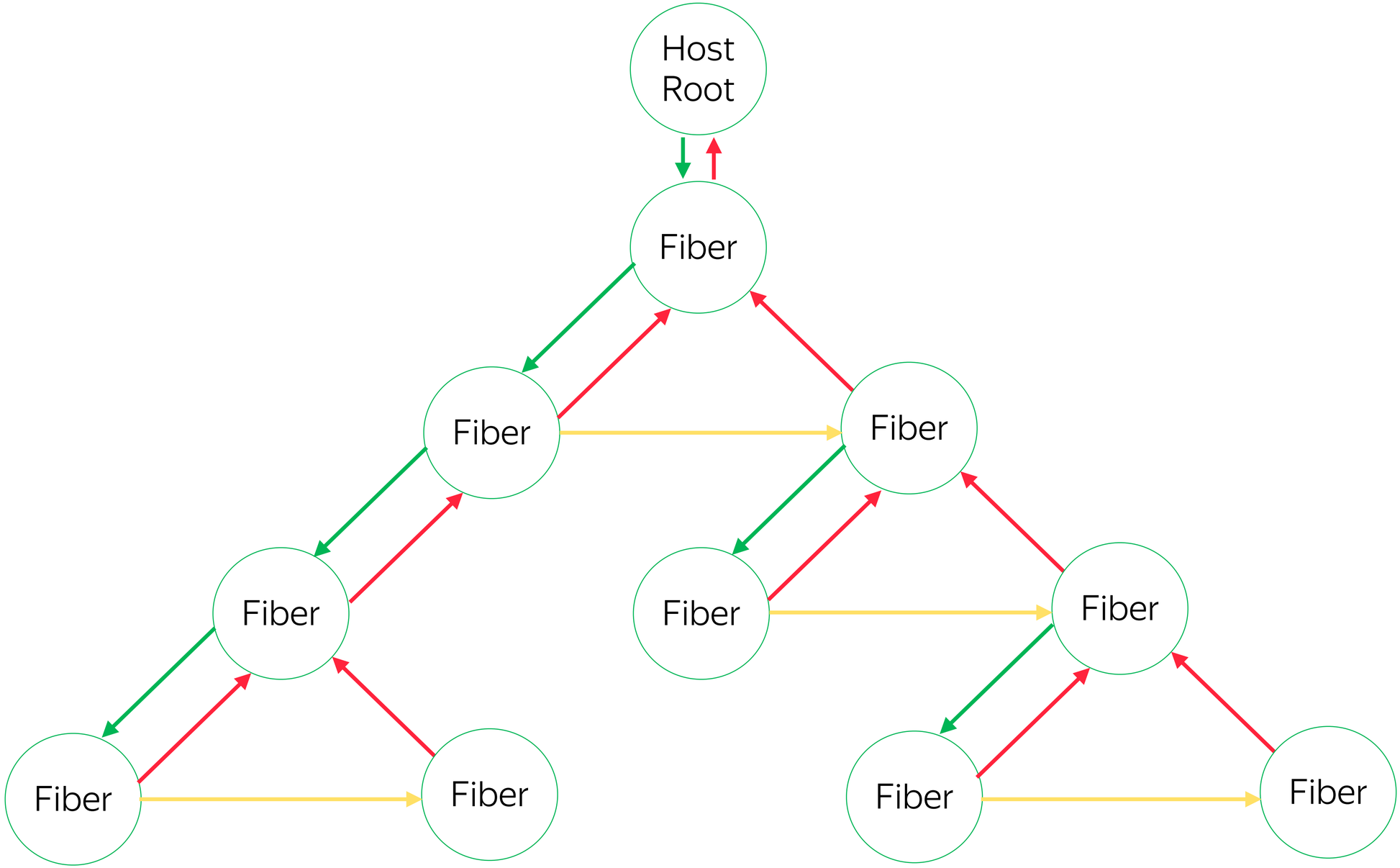
Fiber - Linked List



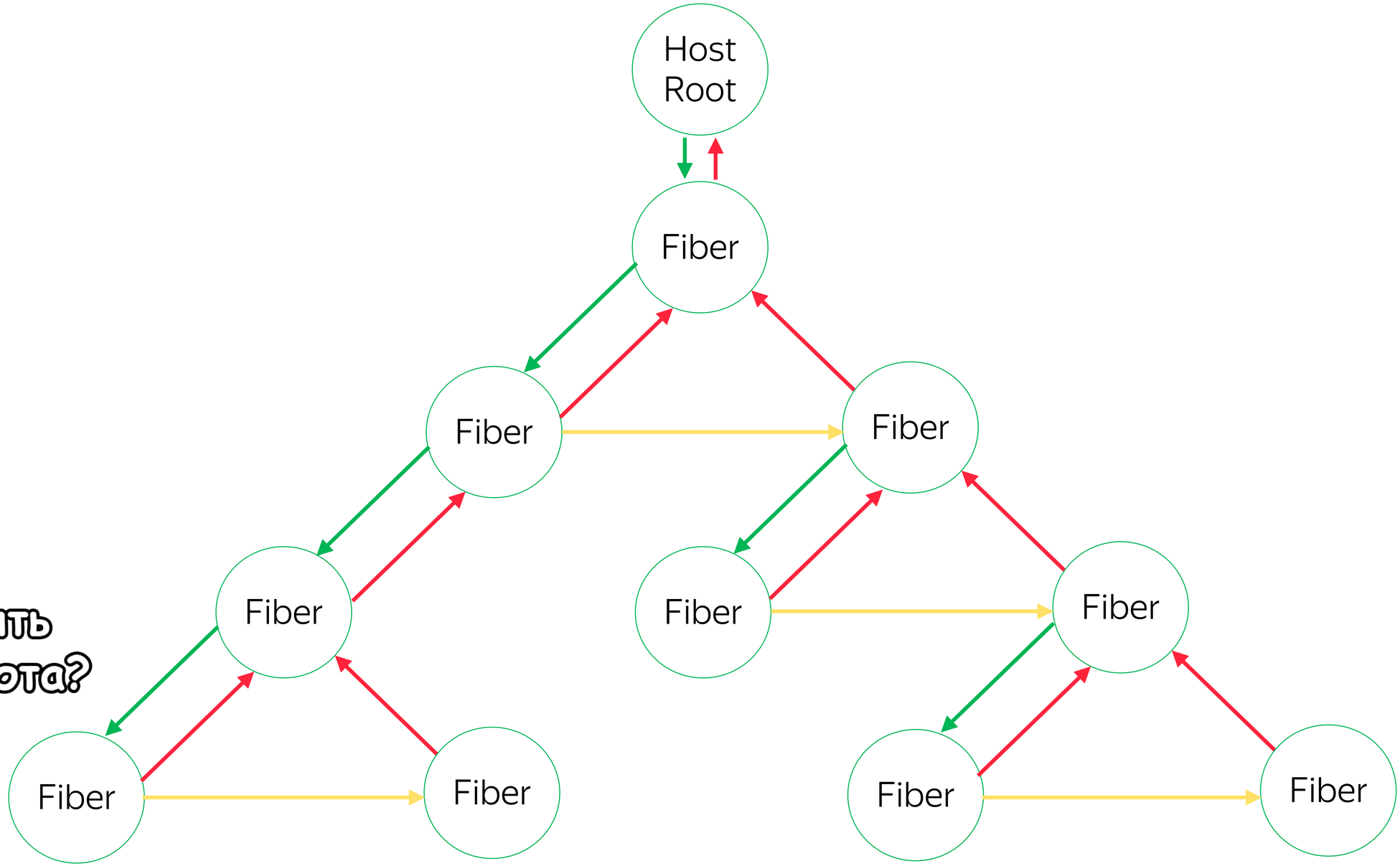
Fiber - Linked List



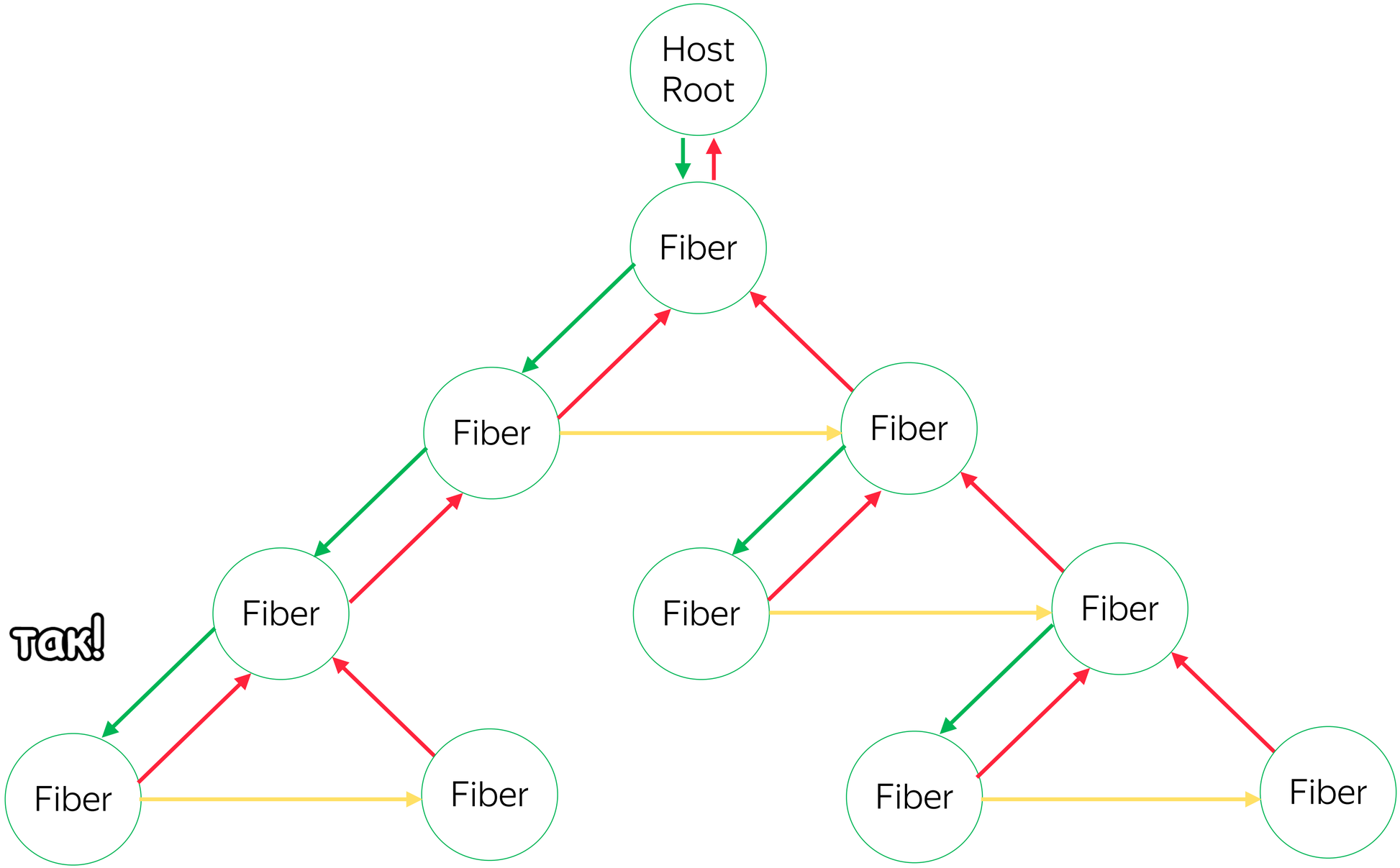
А как это показать пользователю?



Нужно выполнить работу

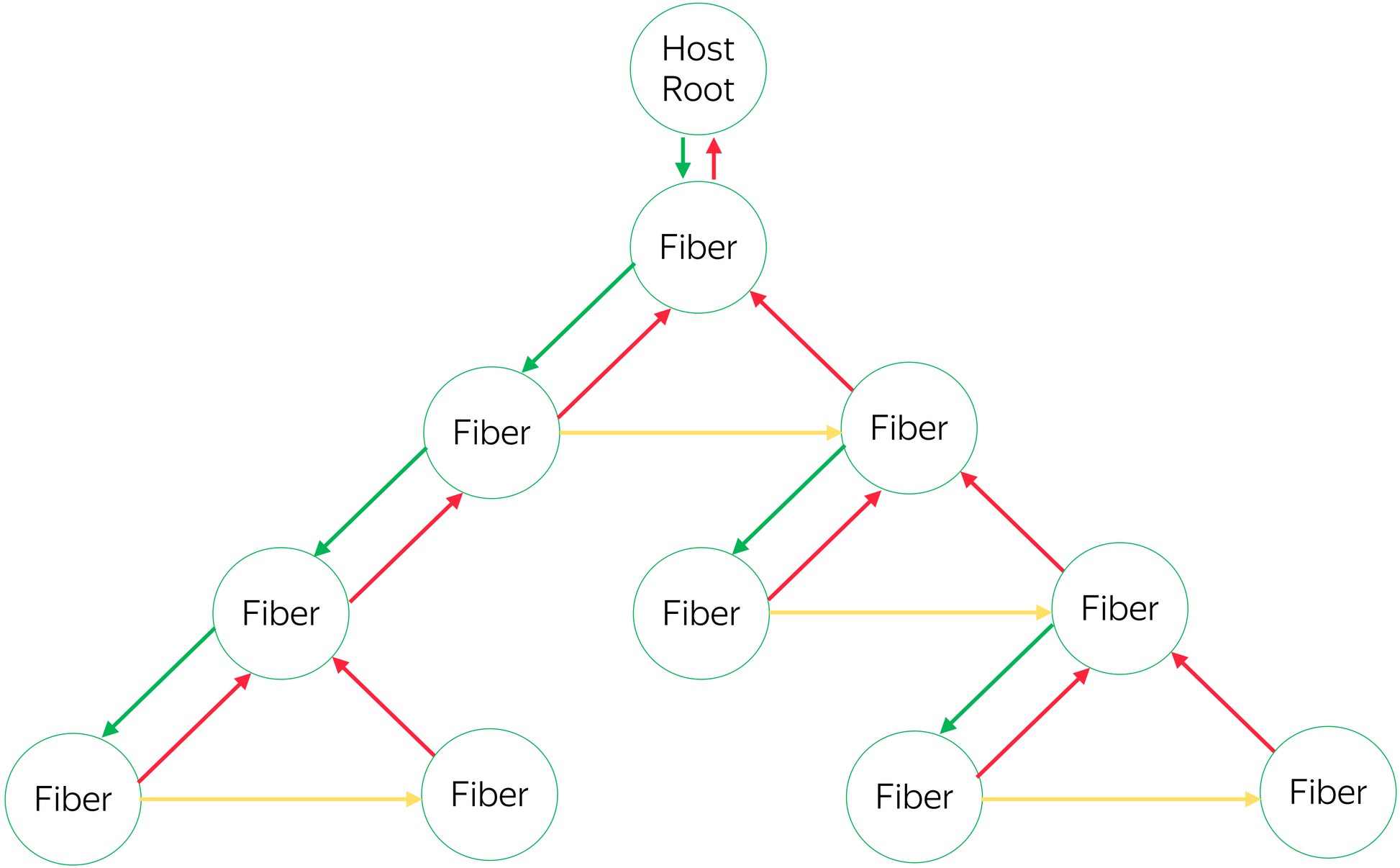


Работа!

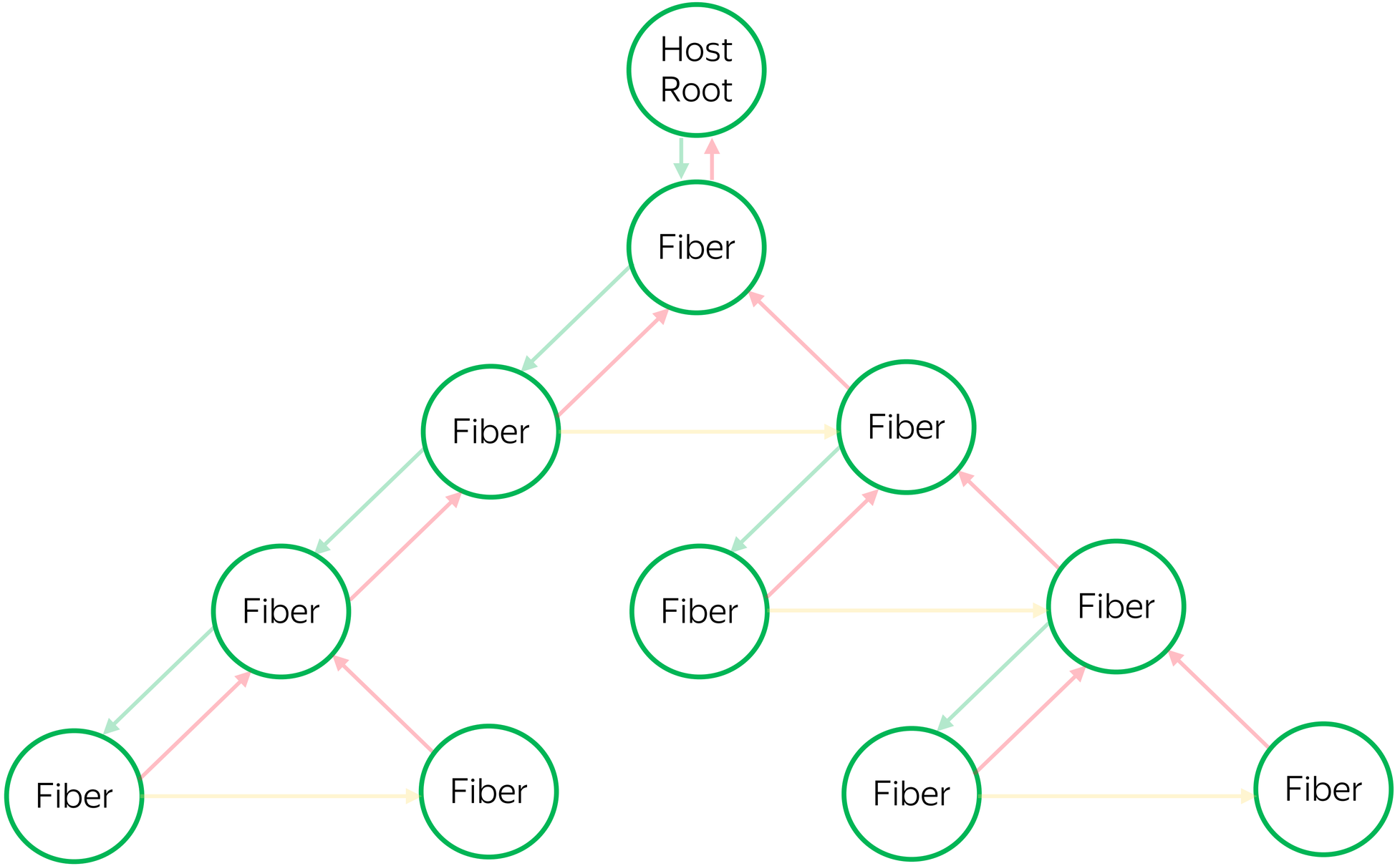


Истинно так!

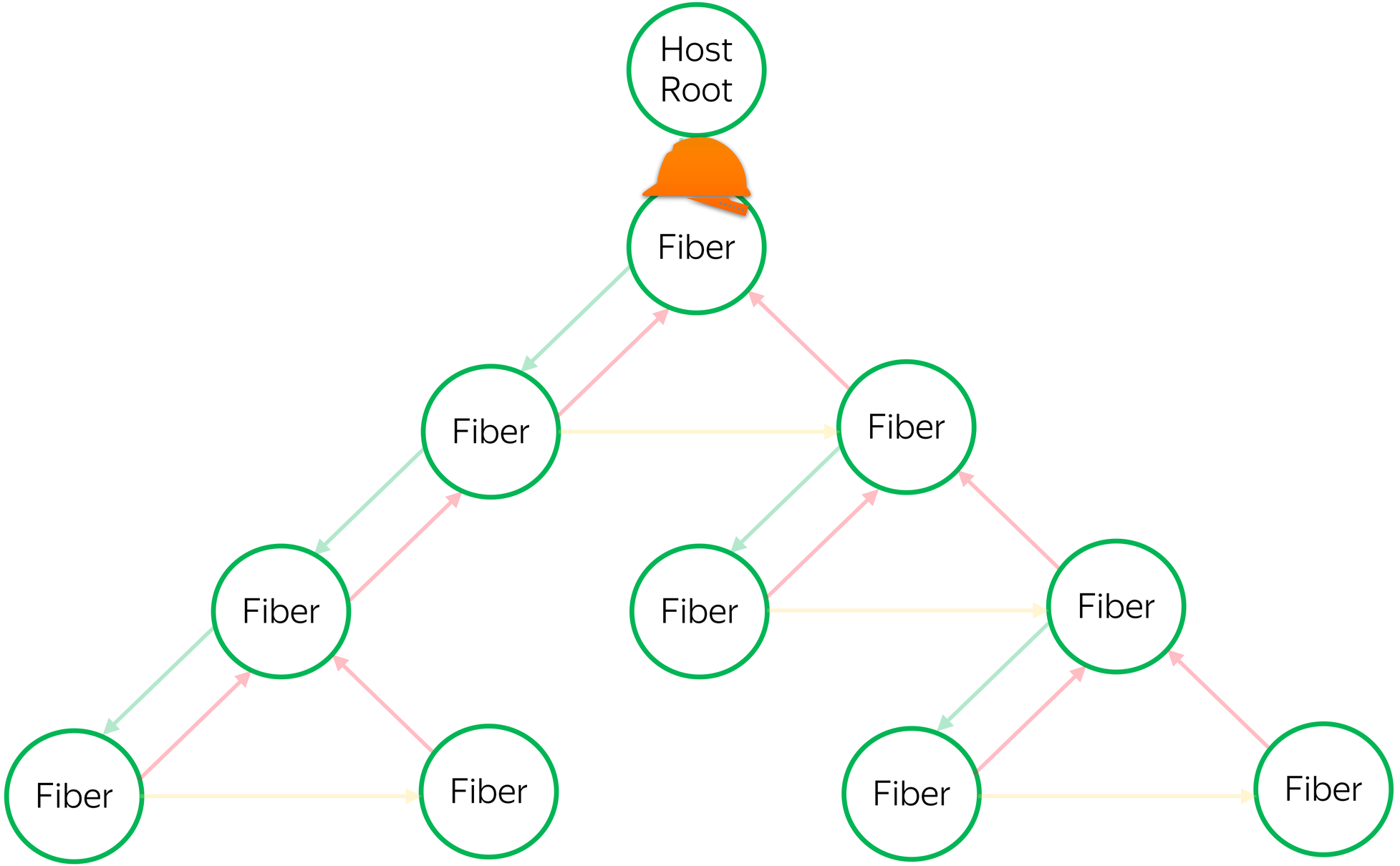
Работа!


Host
Root
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

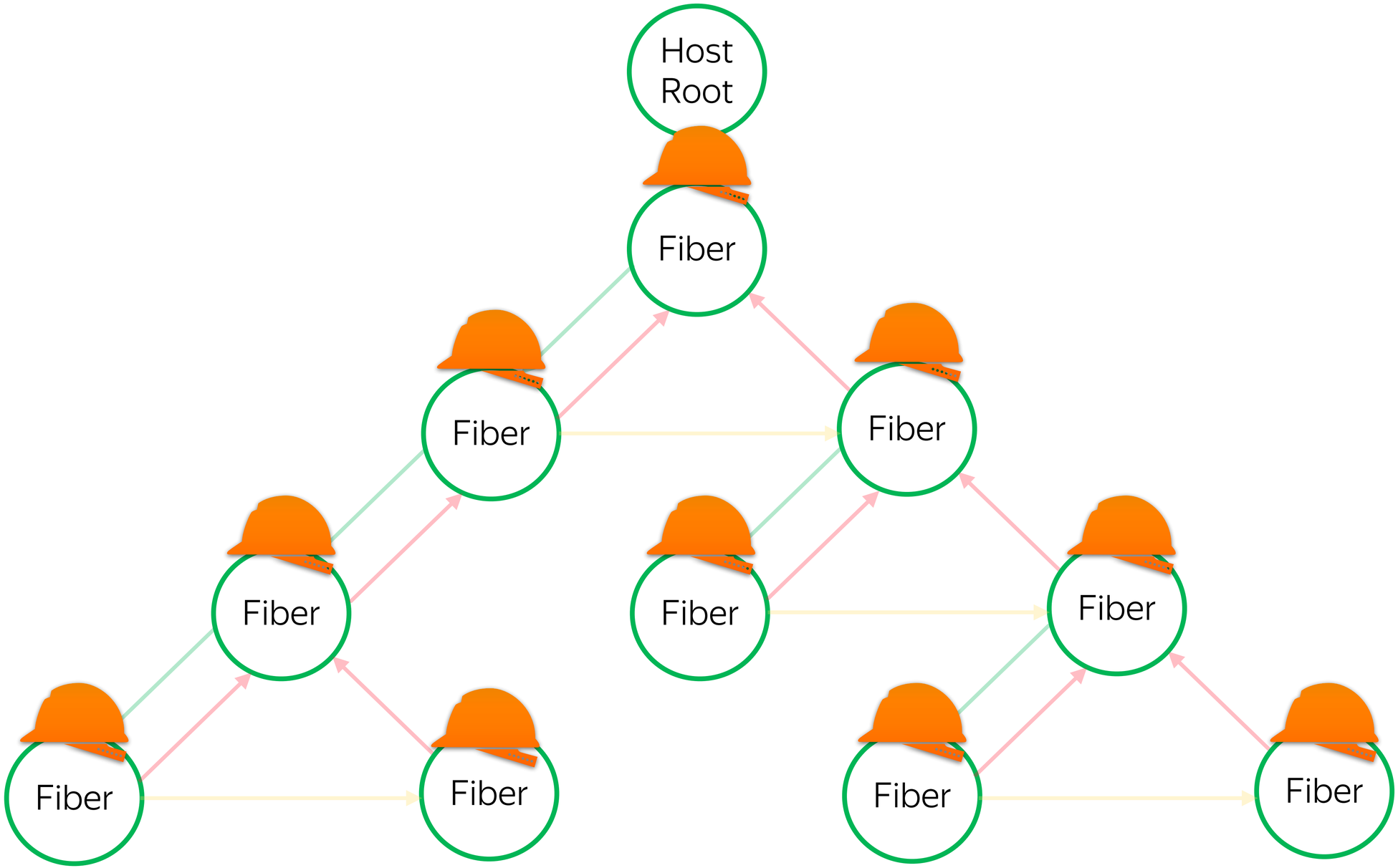
Работа!


Host
Root
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

Работа!


Host
Root
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

Работа!


Host
Root
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

Работа!

Effect!

Effect!

You've likely performed data fetching, subscriptions, or manually changing the DOM from React components before. We call these operations "side effects" (or "effects" for short) because they can affect other components and can't be done during rendering.

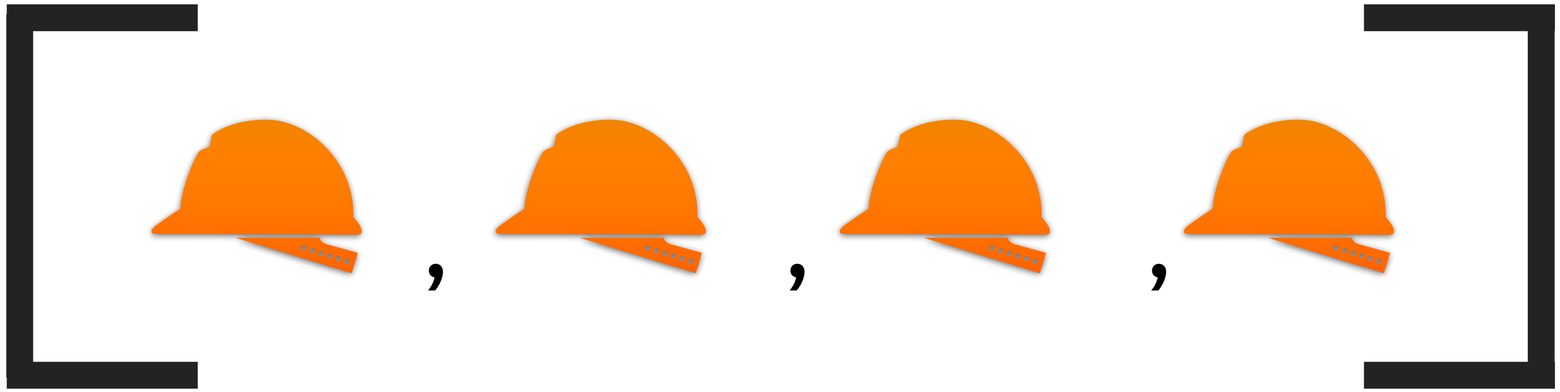
Effect!

You've likely performed data fetching, subscriptions, or manually changing the DOM from React components before. We call these operations "side effects" (or "effects" for short) because they can affect other components and can't be done during rendering.

Effect!

Effects - data fetching, subscriptions, or manually changing the DOM

Effect - list!



Effect - list!

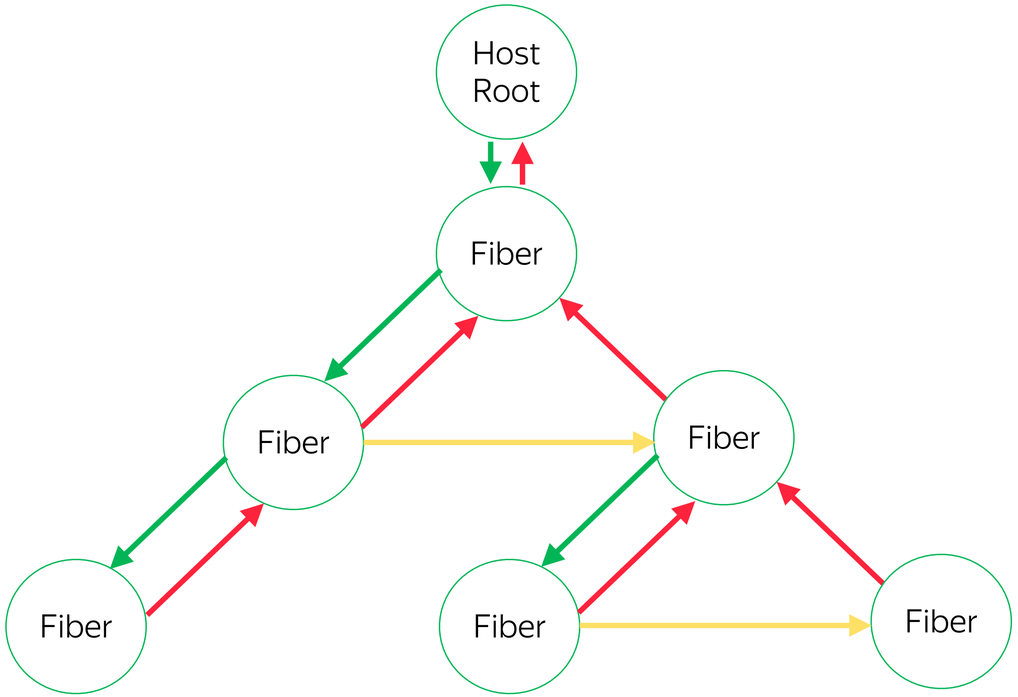
Effect - list!

Фаза 2
Commit

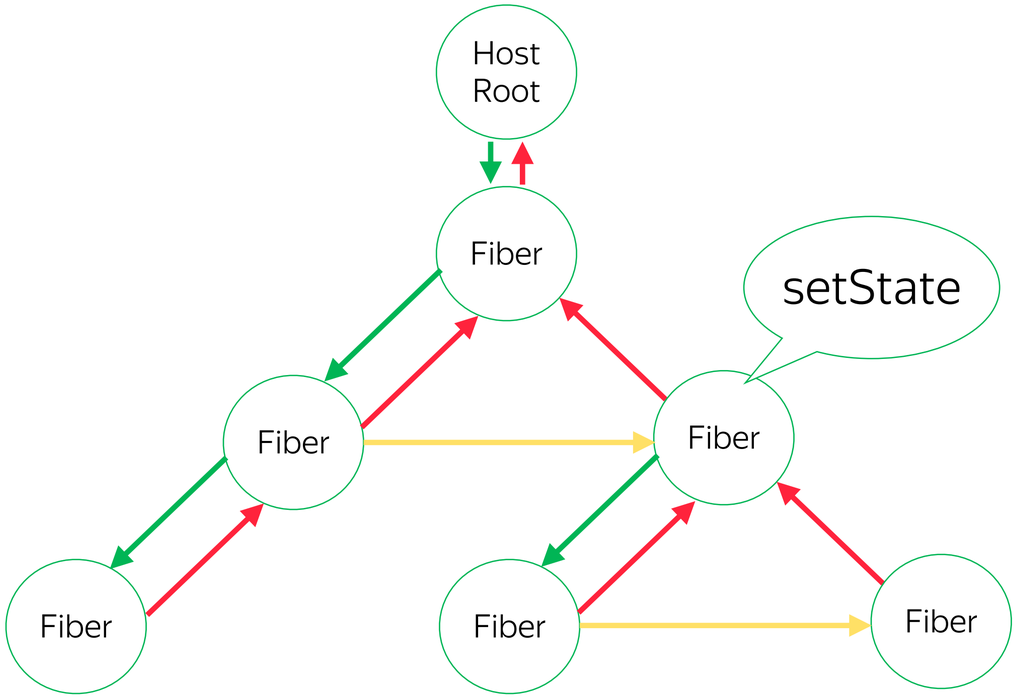
Подождите!
А обновления???

Фаза 1
Рендеринг и Сравнение
(Продолжение)

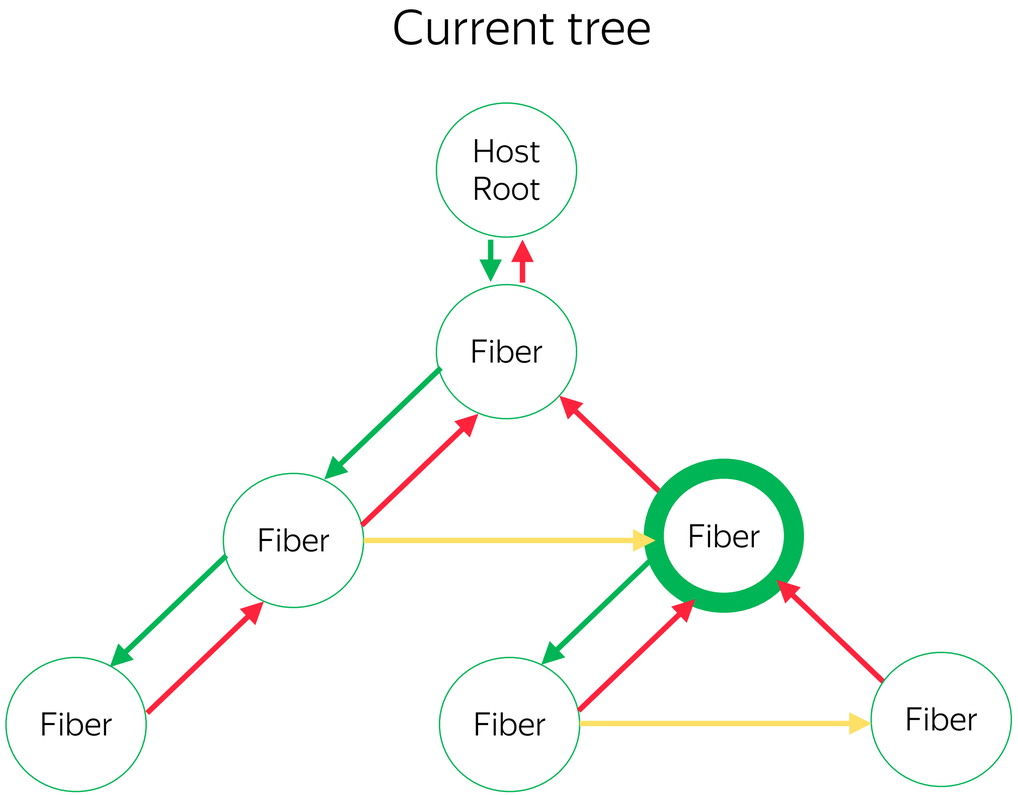
Current tree!



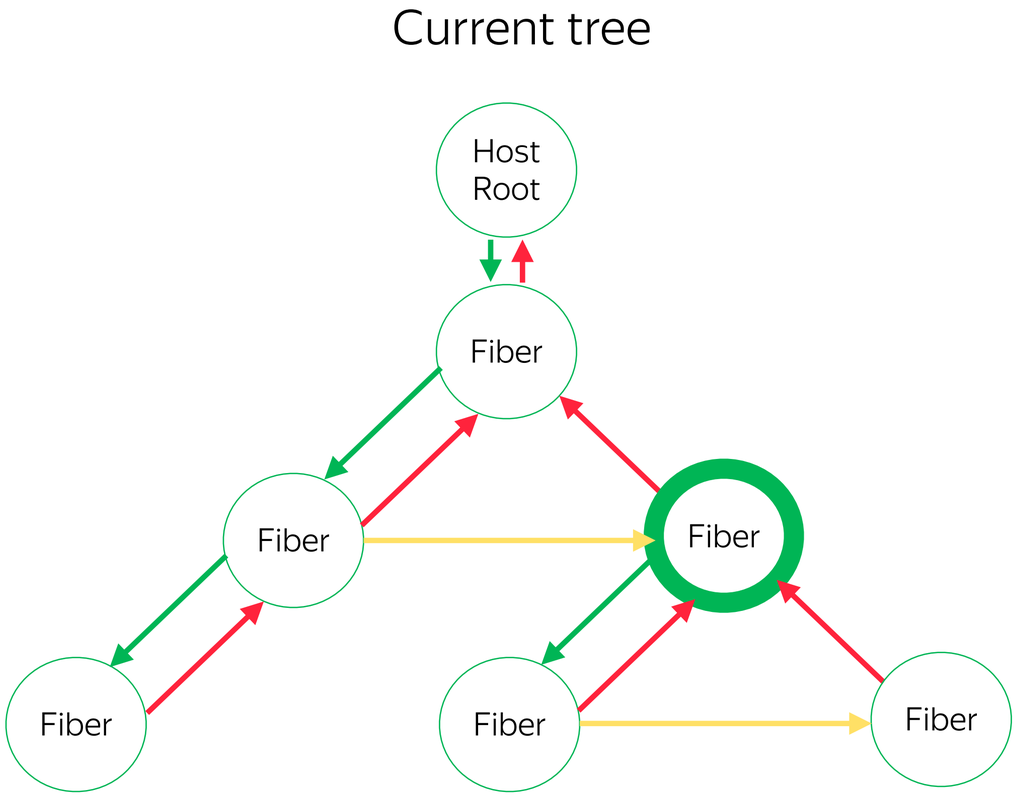
Need re-render!



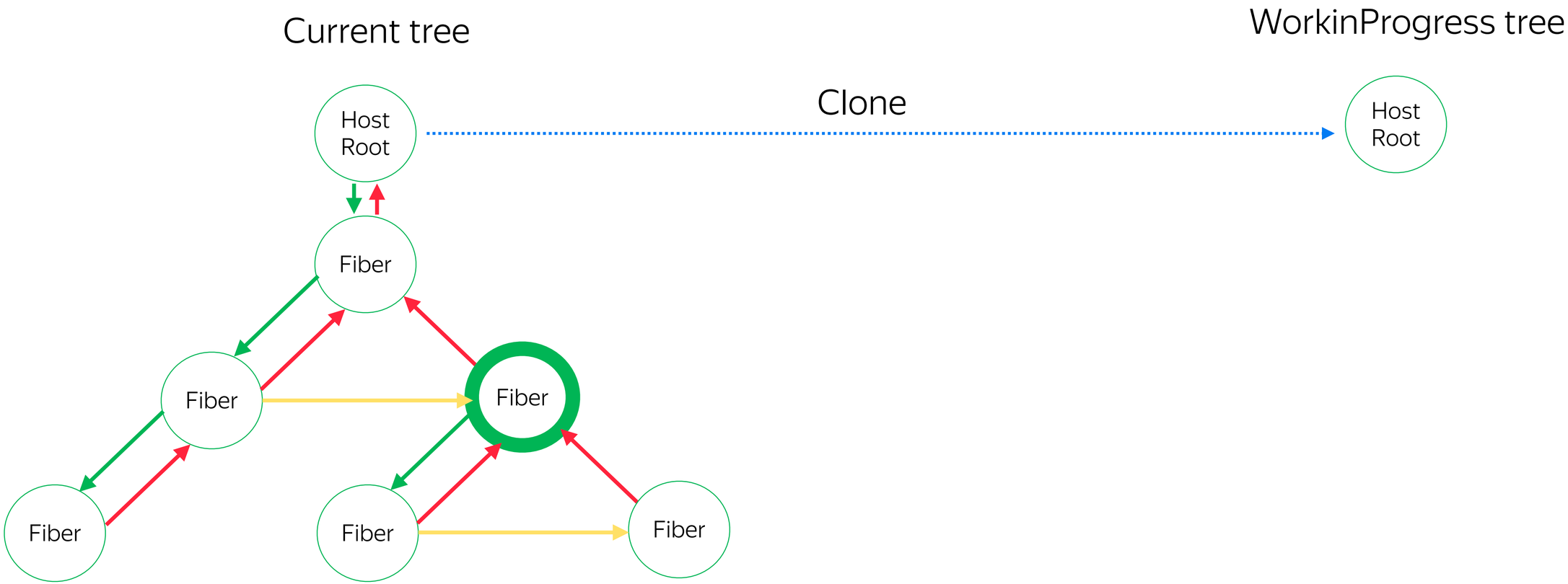
New tree?



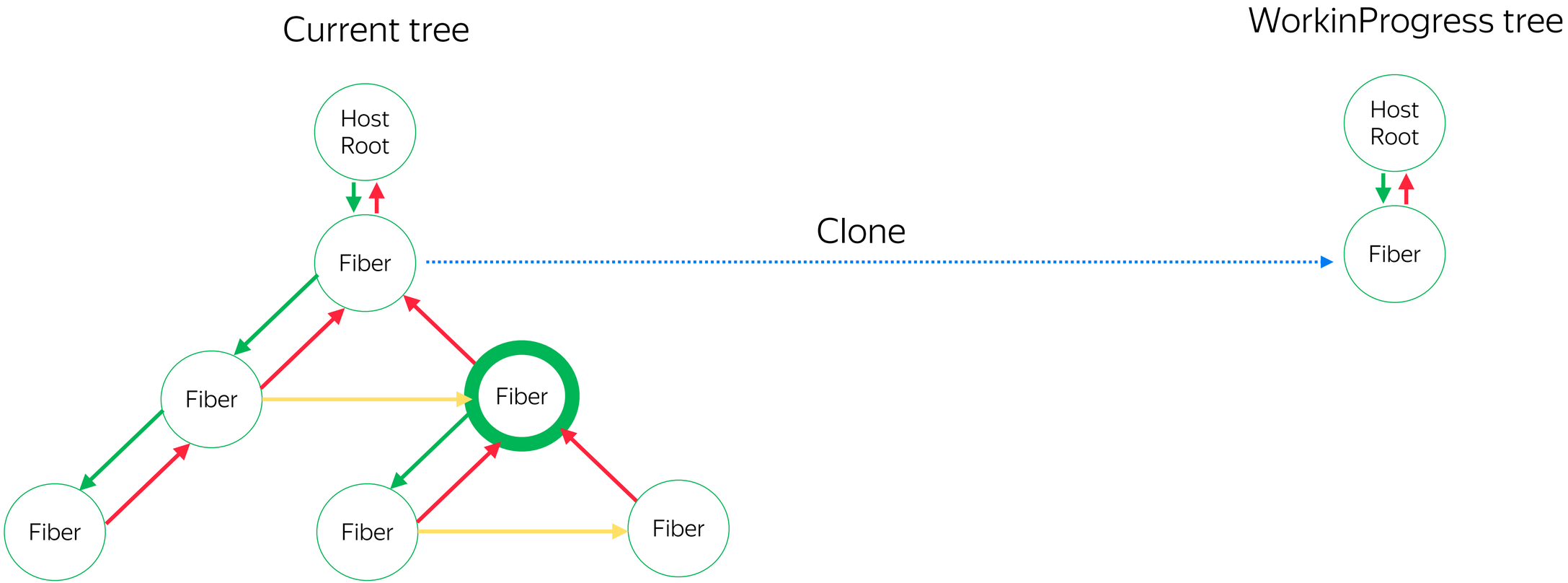
New tree? Yes!



New tree? Yes!

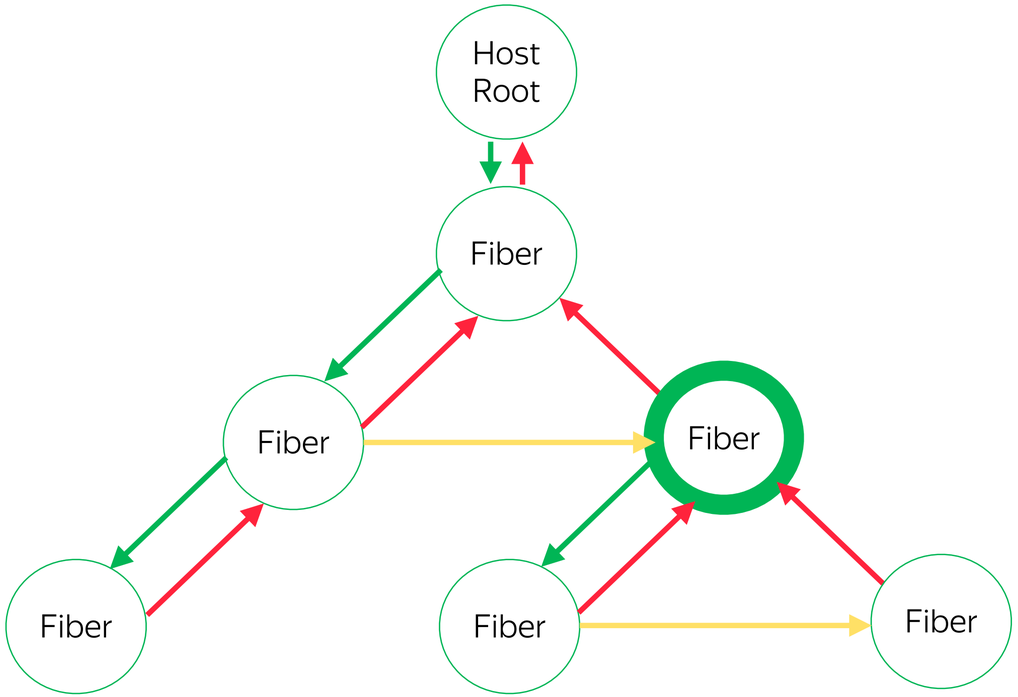

Current tree
WorkinProgress tree
Clone
Host Root
Host Root
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

New tree? Yes!

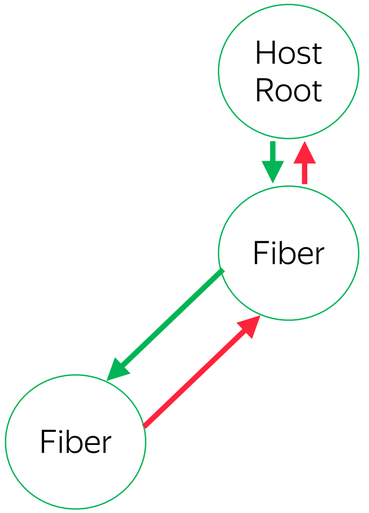

Current tree
WorkinProgress tree
Host Root
Fiber
Clone
Host Root
Fiber
Fiber
Fiber
Fiber
Fiber
Fiber

New tree? Yes!

Current tree

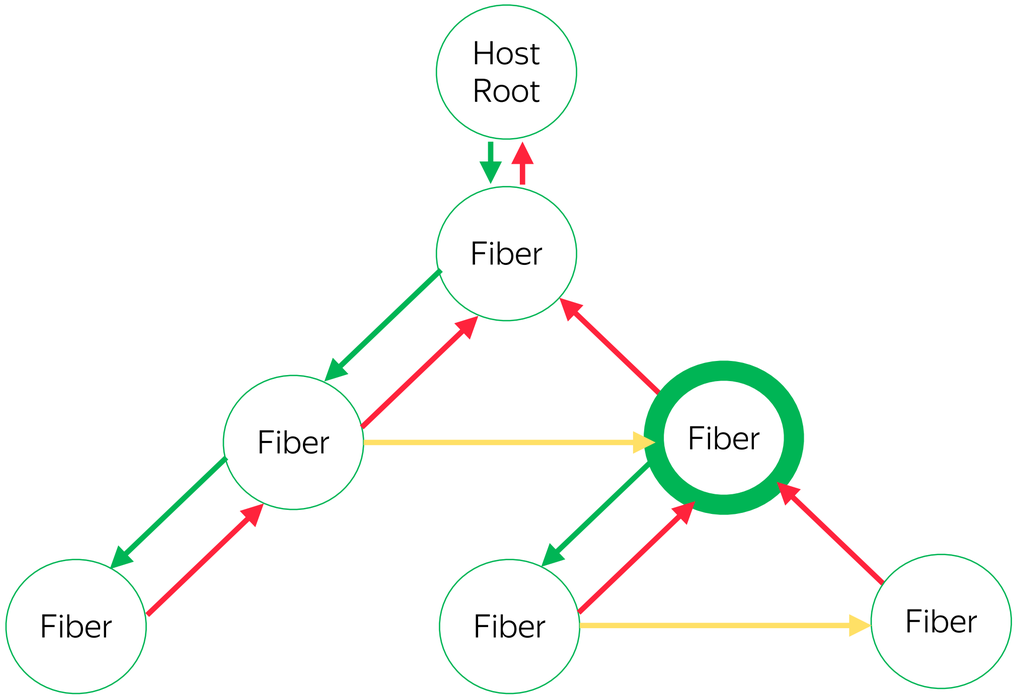


WorkinProgress tree

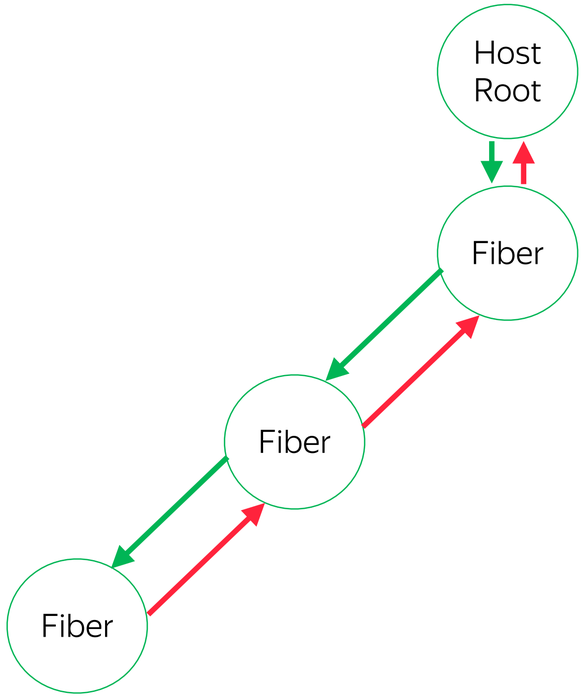


New tree? Yes!

Current tree

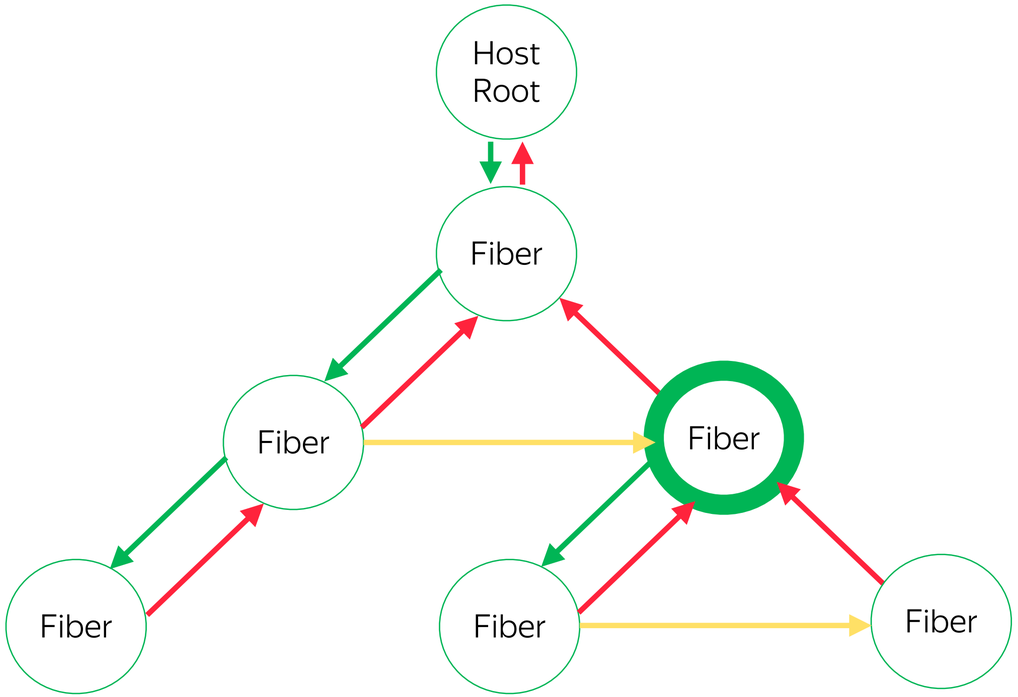


WorkinProgress tree

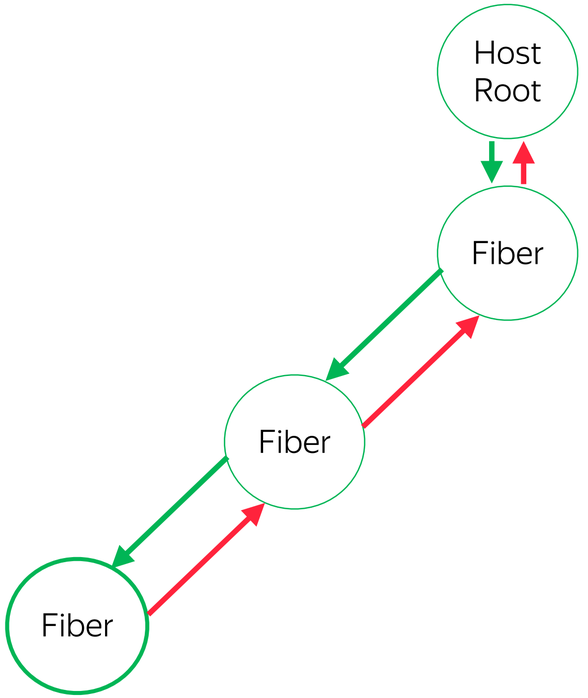


New tree? Yes!

Current tree

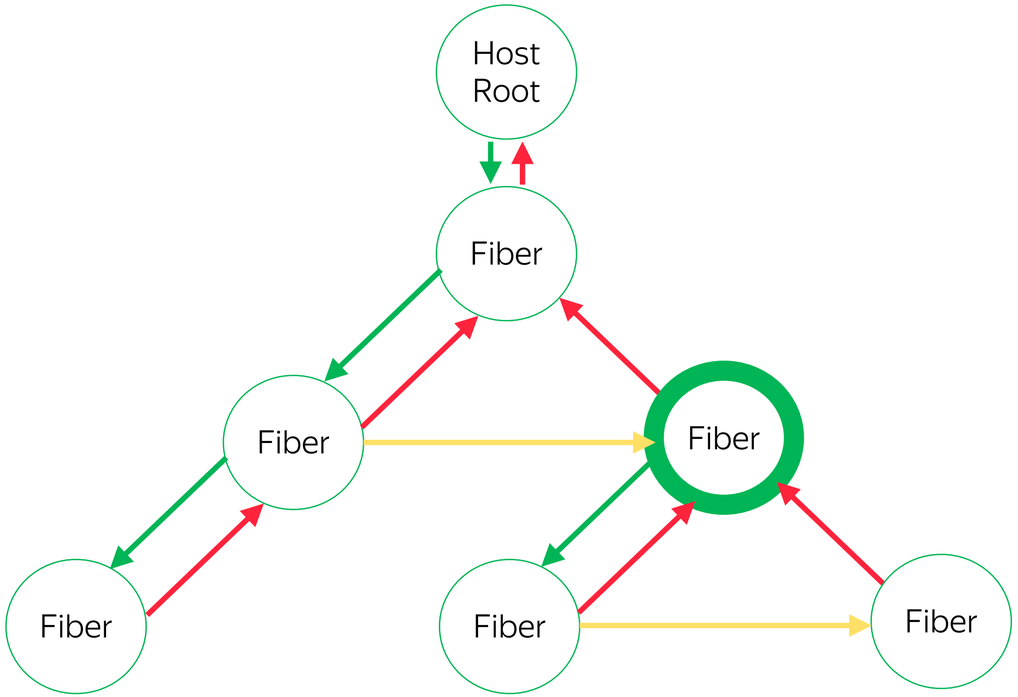


WorkinProgress tree

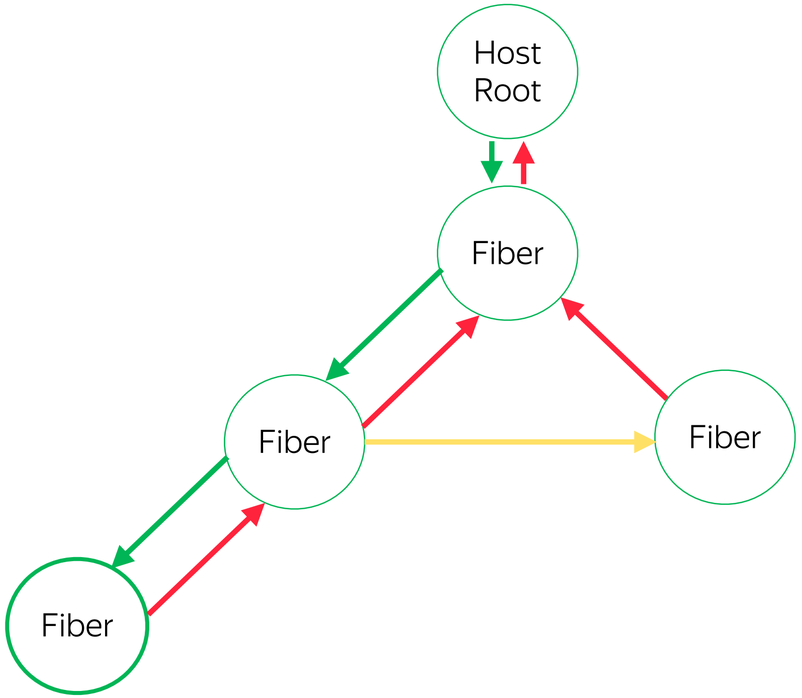


New tree? Yes!

Current tree

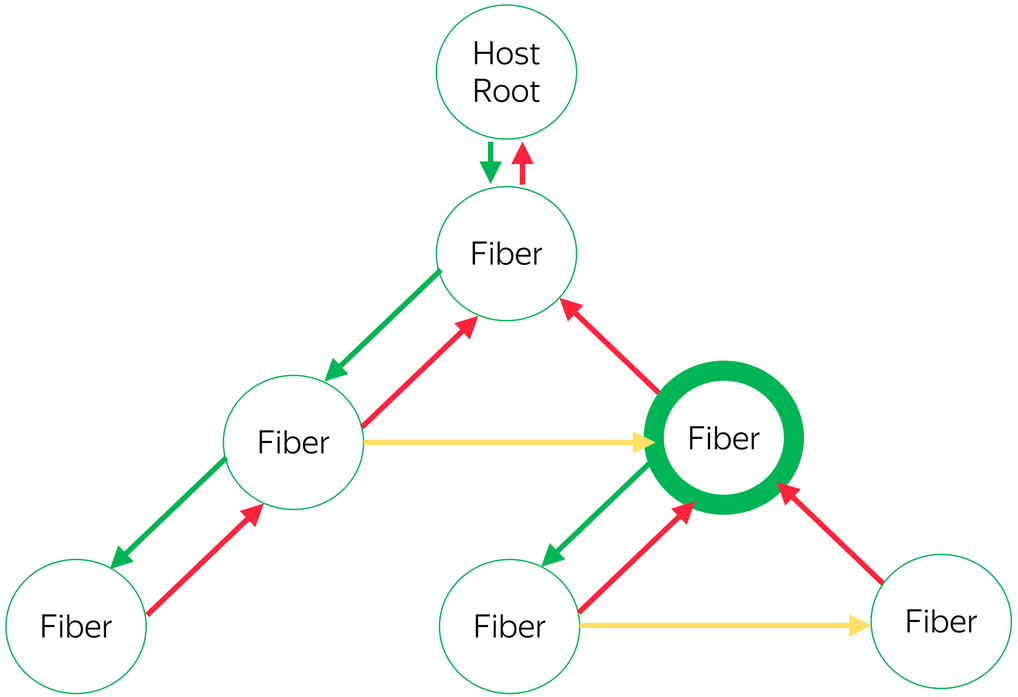


WorkinProgress tree

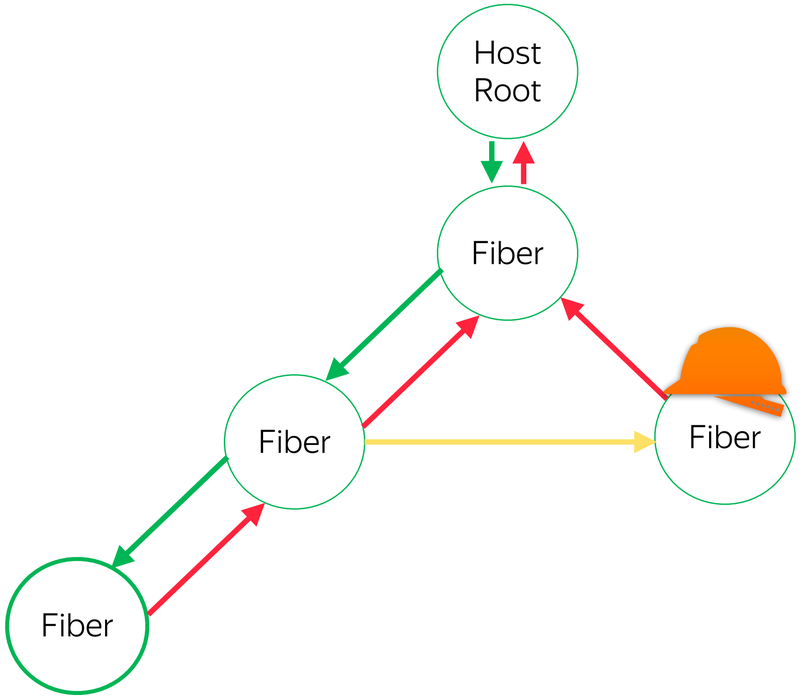


New tree? Yes!

Current tree

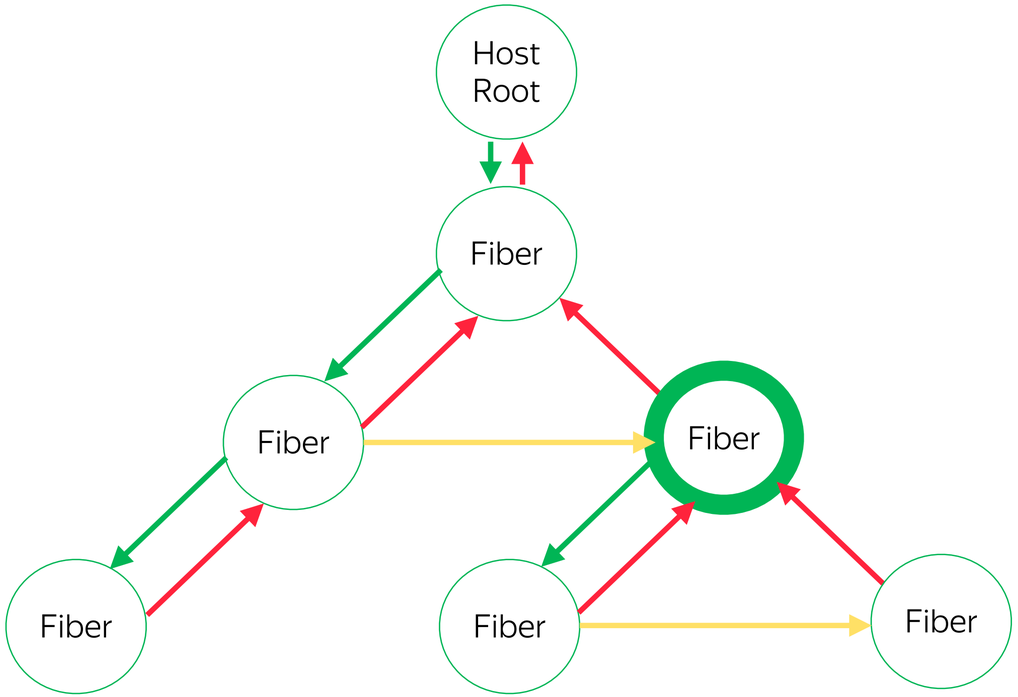


WorkinProgress tree

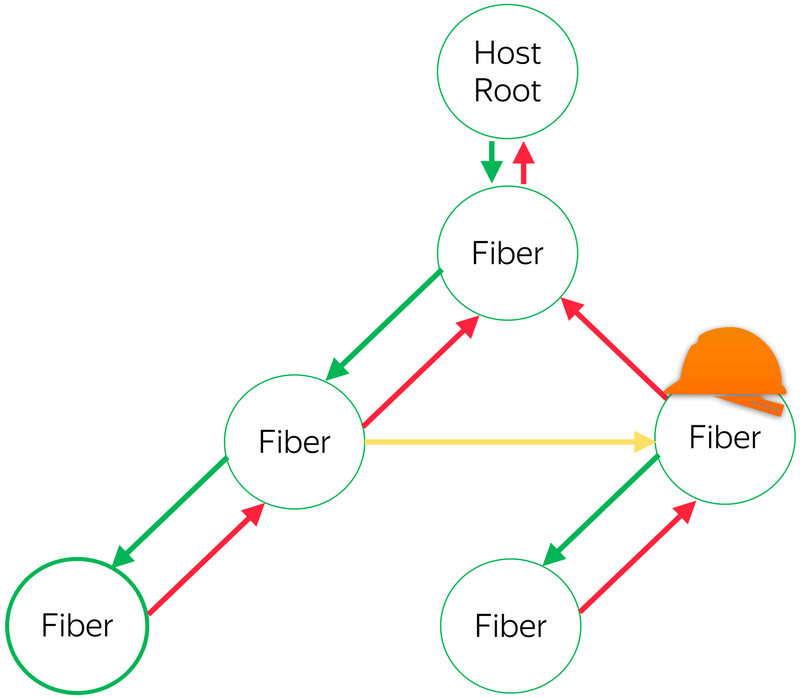


New tree? Yes!

Current tree

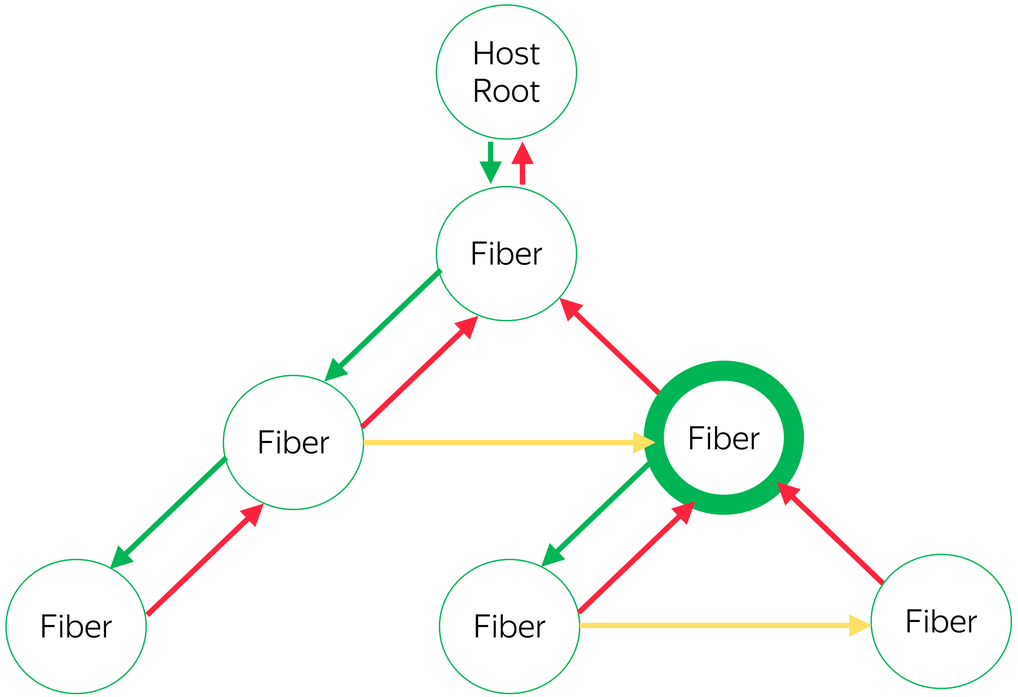


WorkinProgress tree

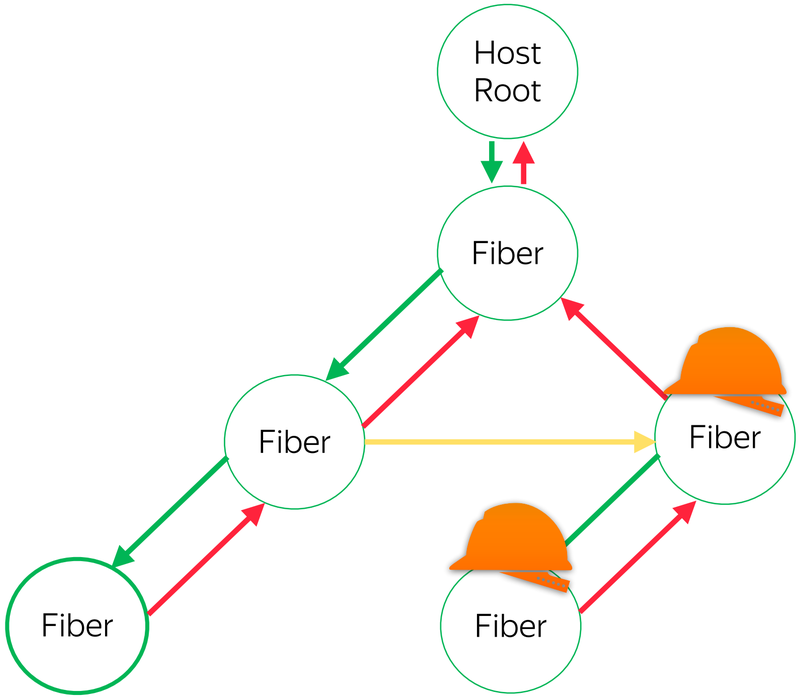


New tree? Yes!

Current tree

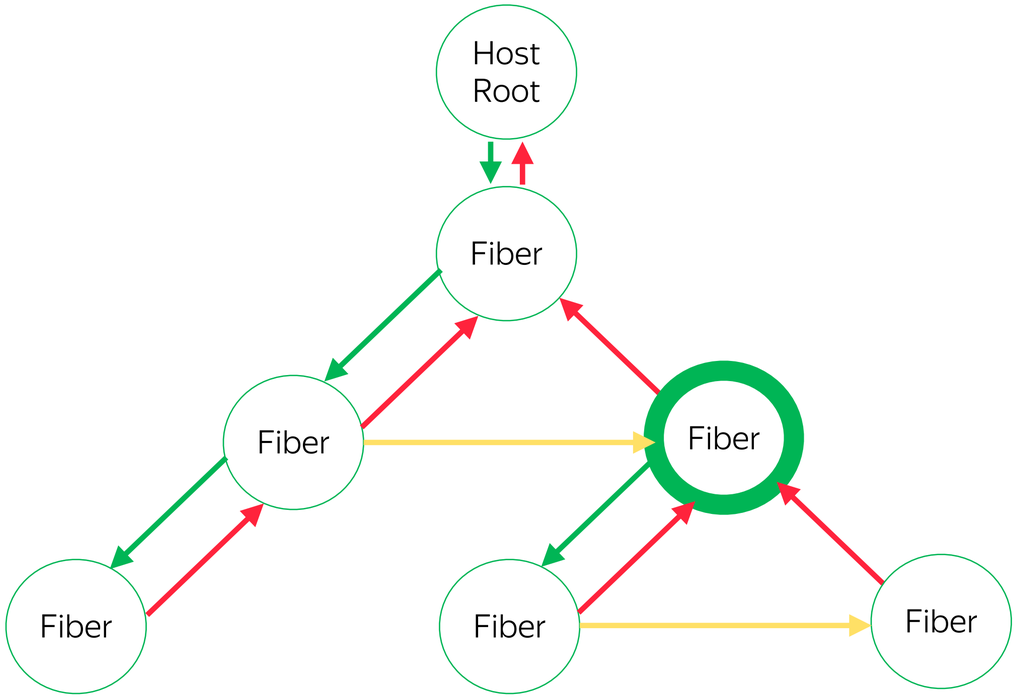


WorkinProgress tree

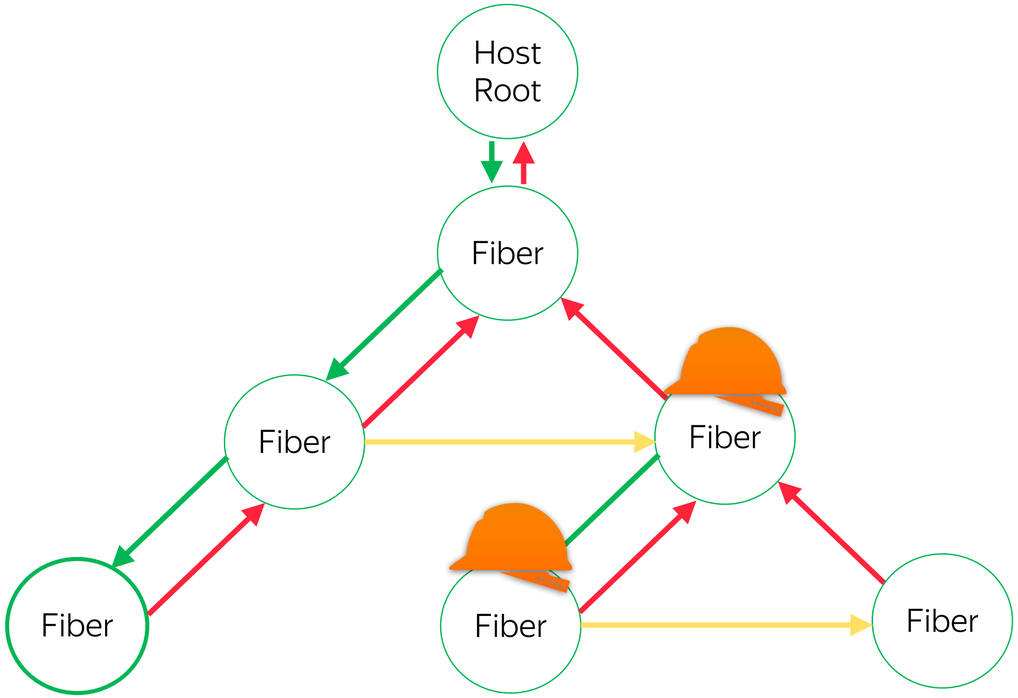


New tree? Yes!

Current tree

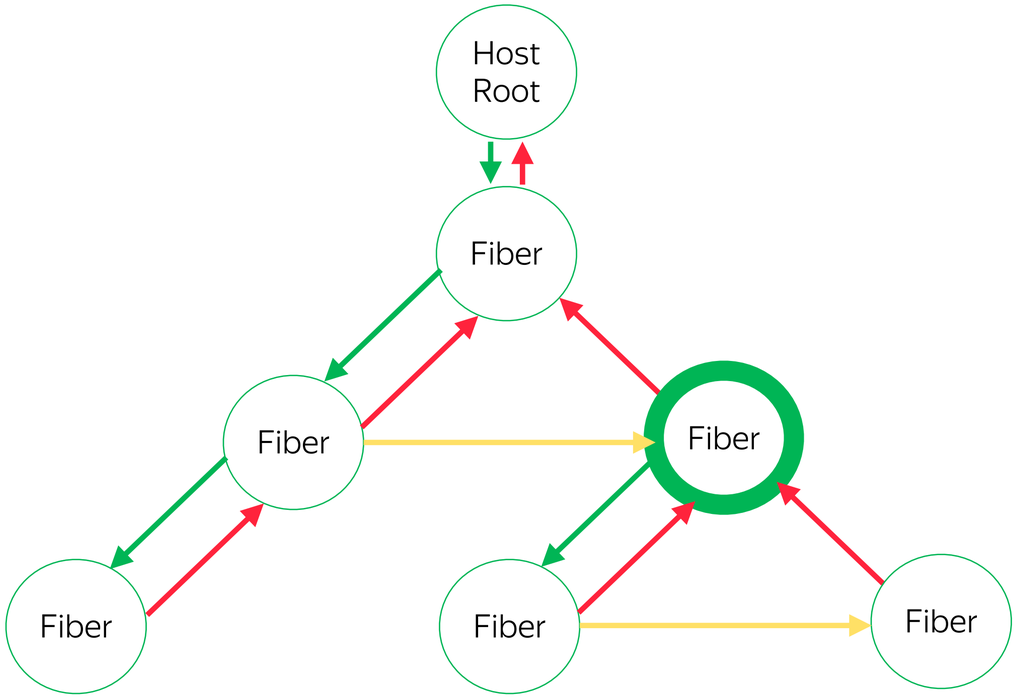


WorkinProgress tree

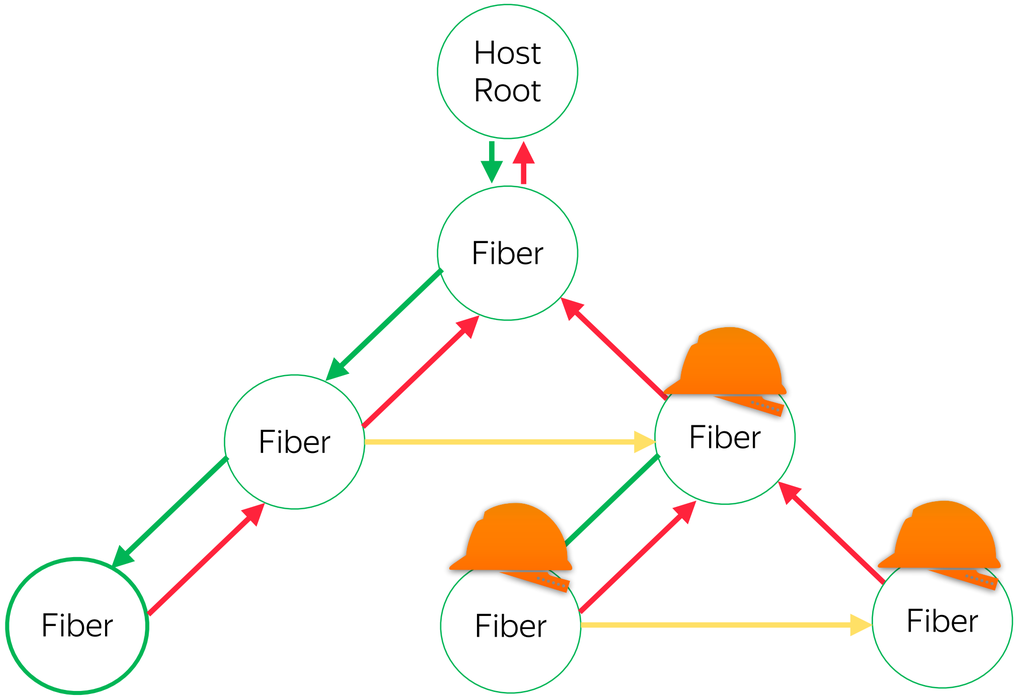


New tree? Yes!

Current tree

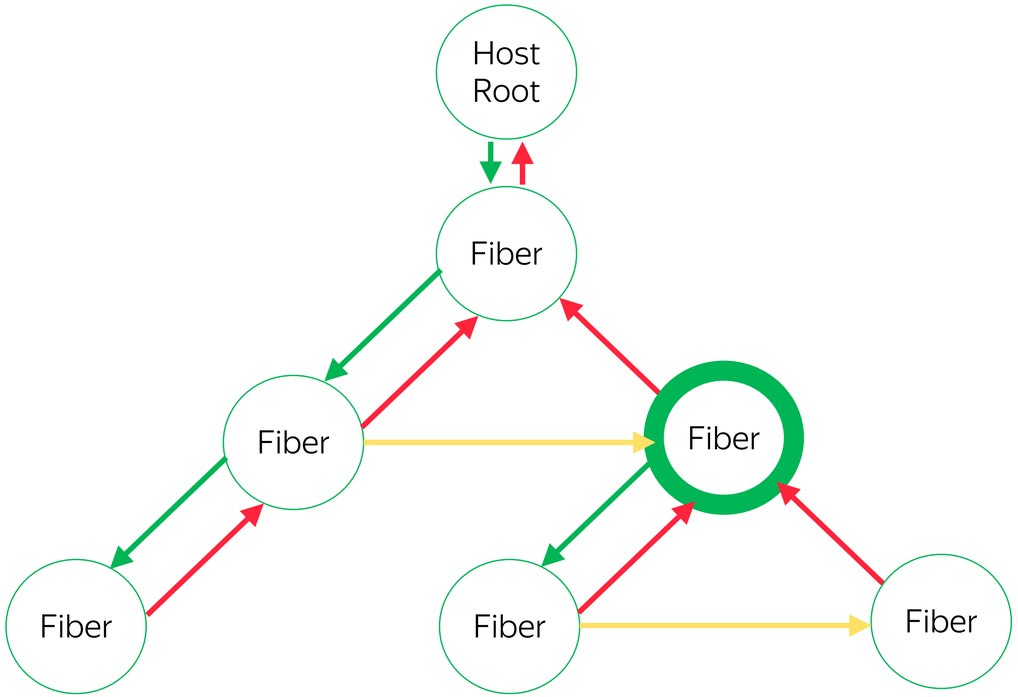


WorkinProgress tree

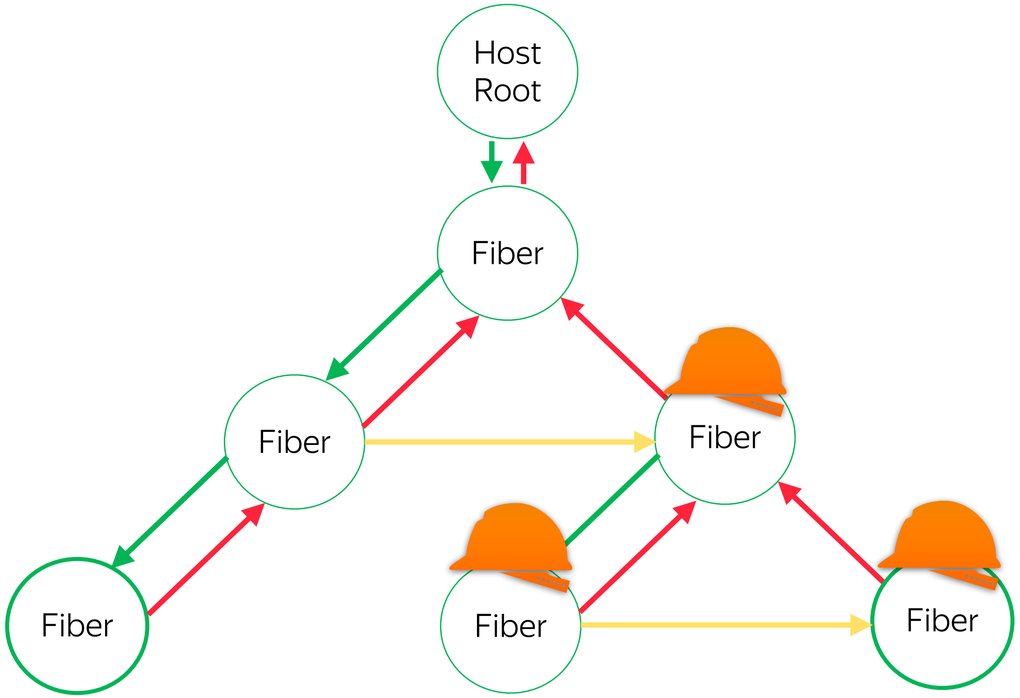


New tree? Yes!

Current tree

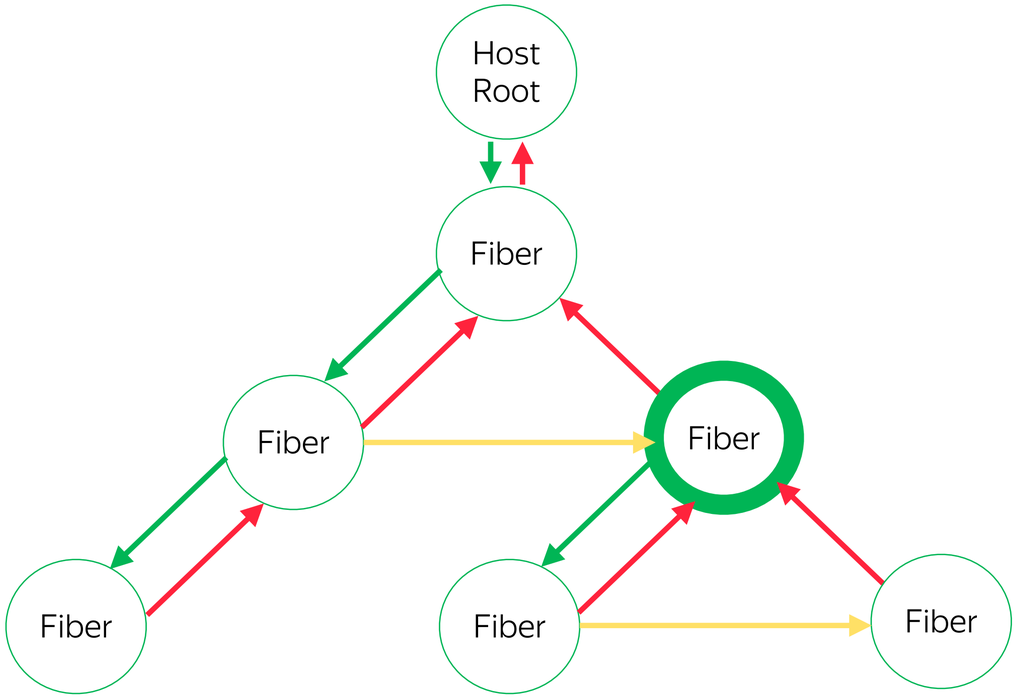


WorkinProgress tree

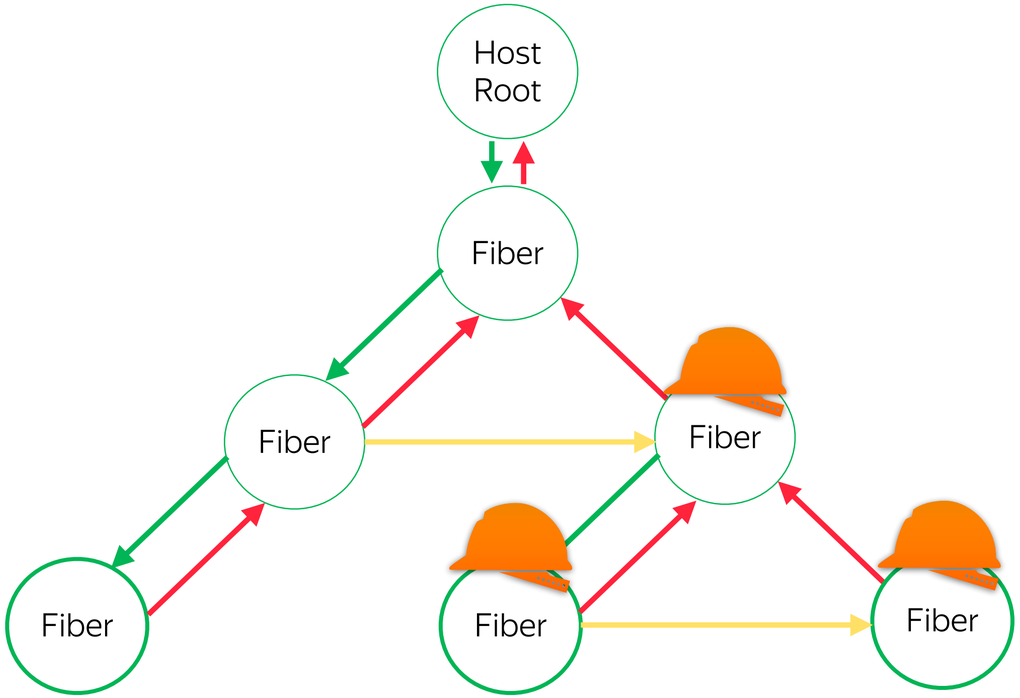


New tree? Yes!

Current tree

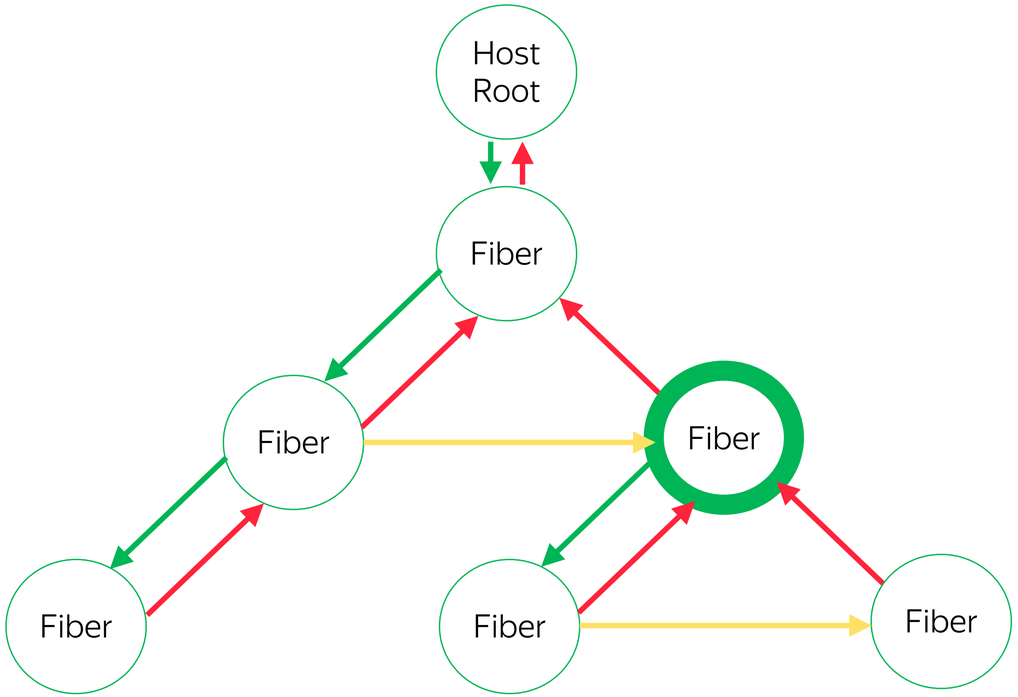


WorkinProgress tree

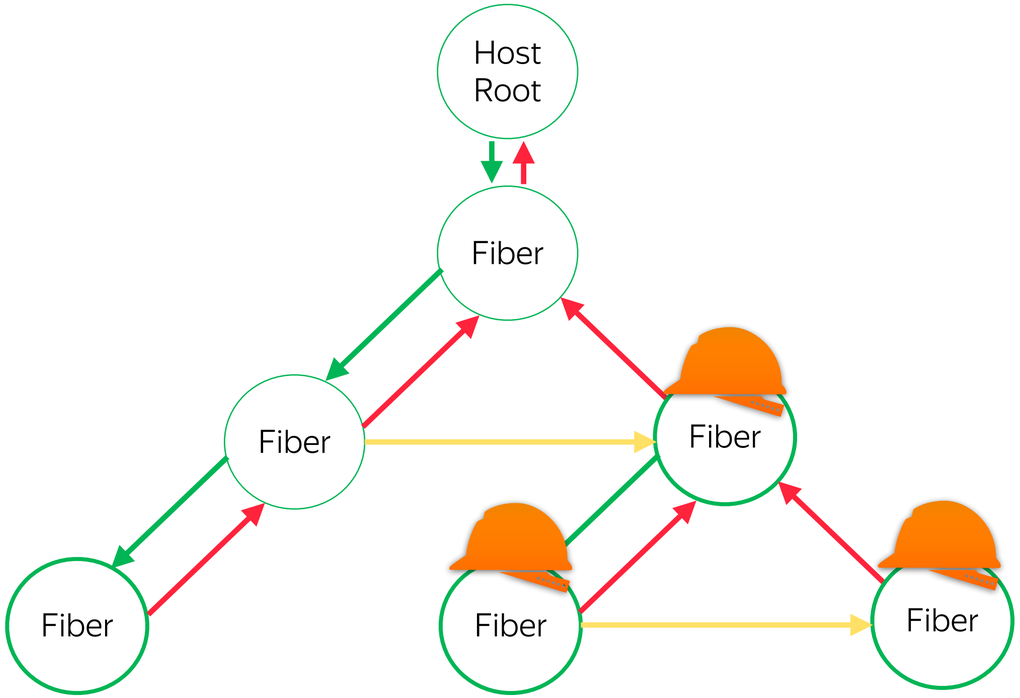


New tree? Yes!

Current tree

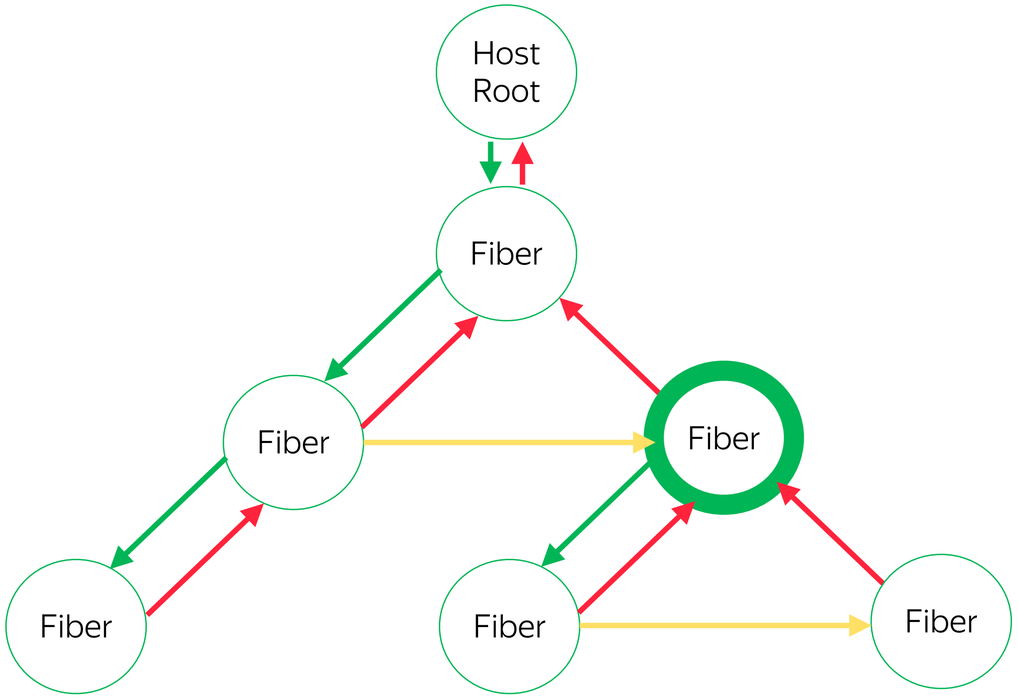


WorkinProgress tree

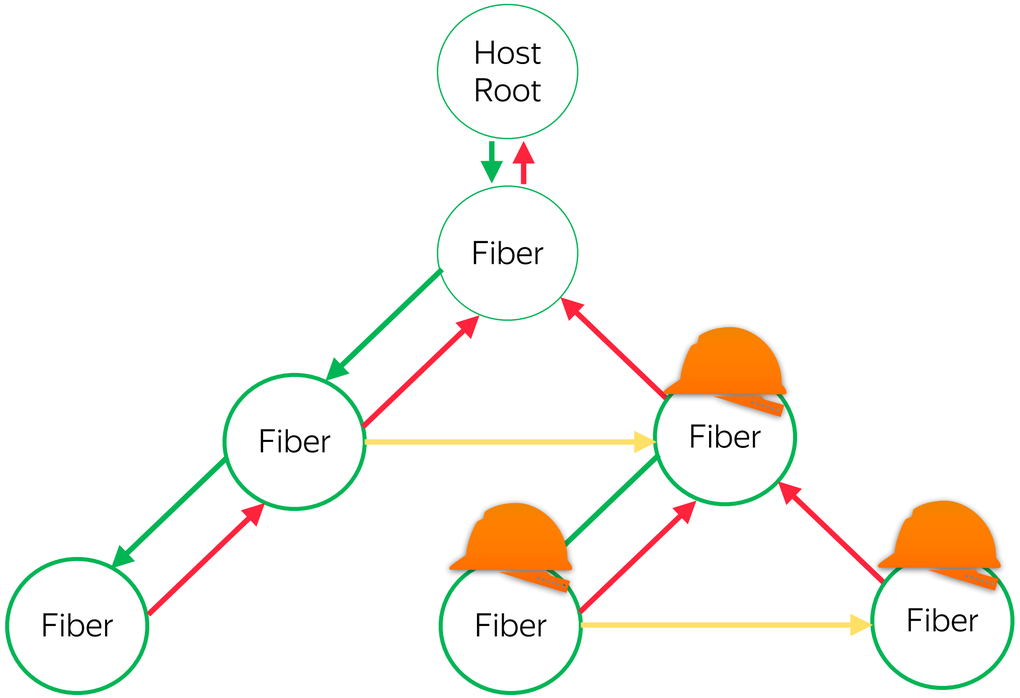


New tree? Yes!

Current tree

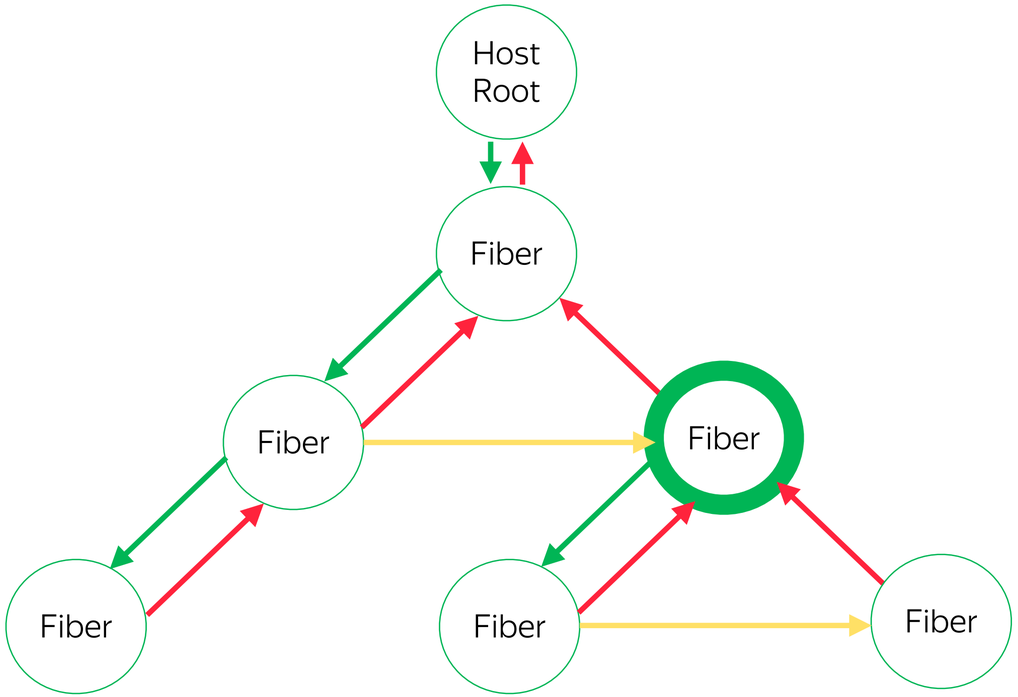


WorkinProgress tree

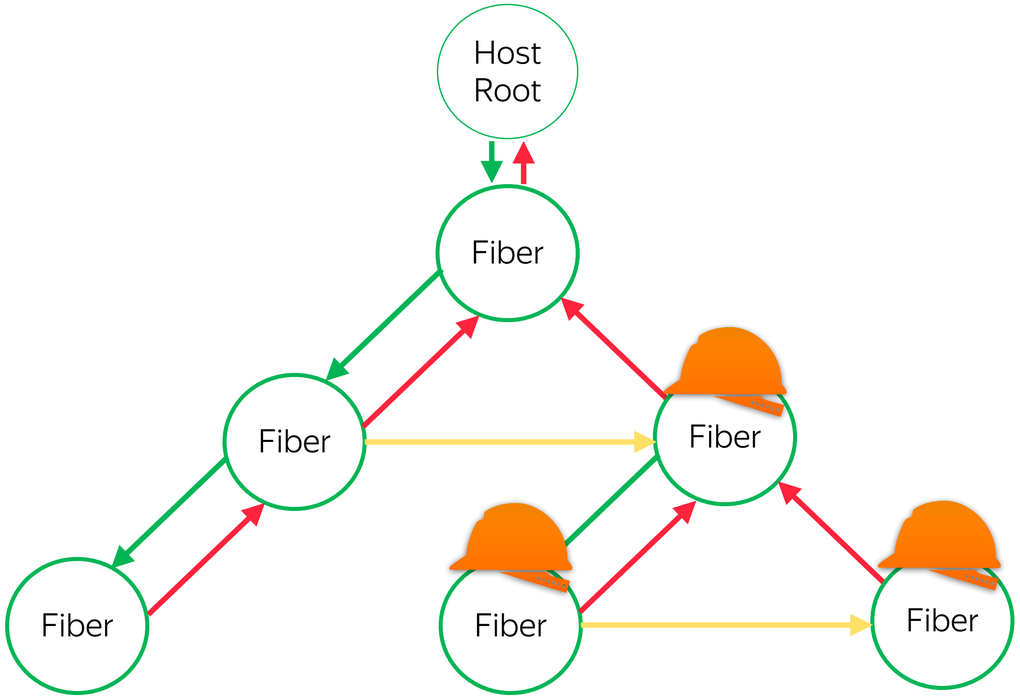


New tree? Yes!

Current tree

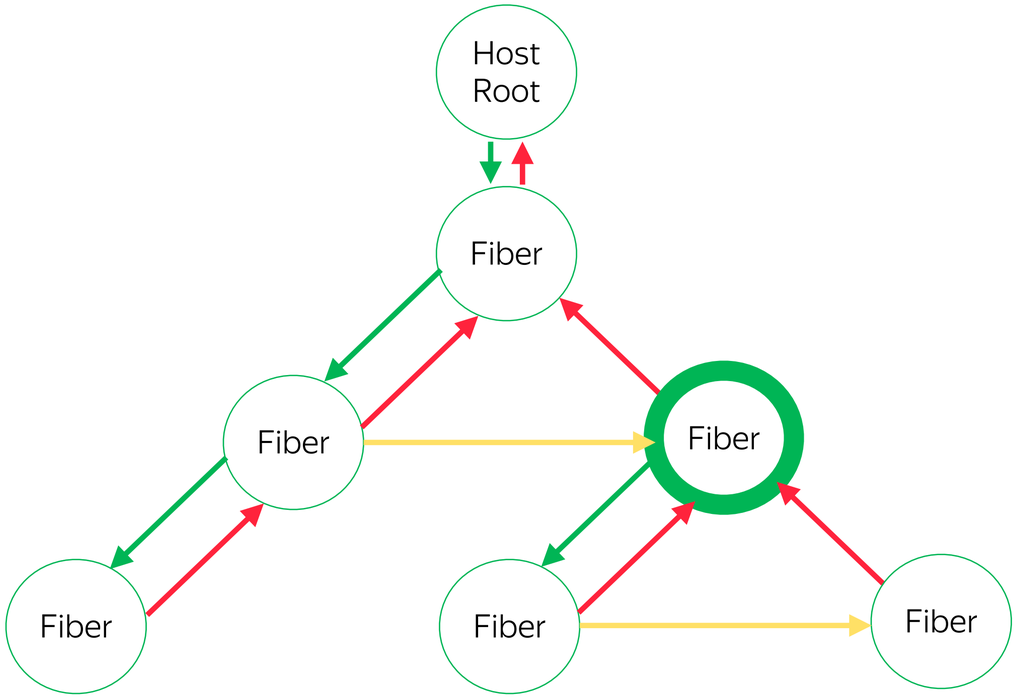


WorkinProgress tree

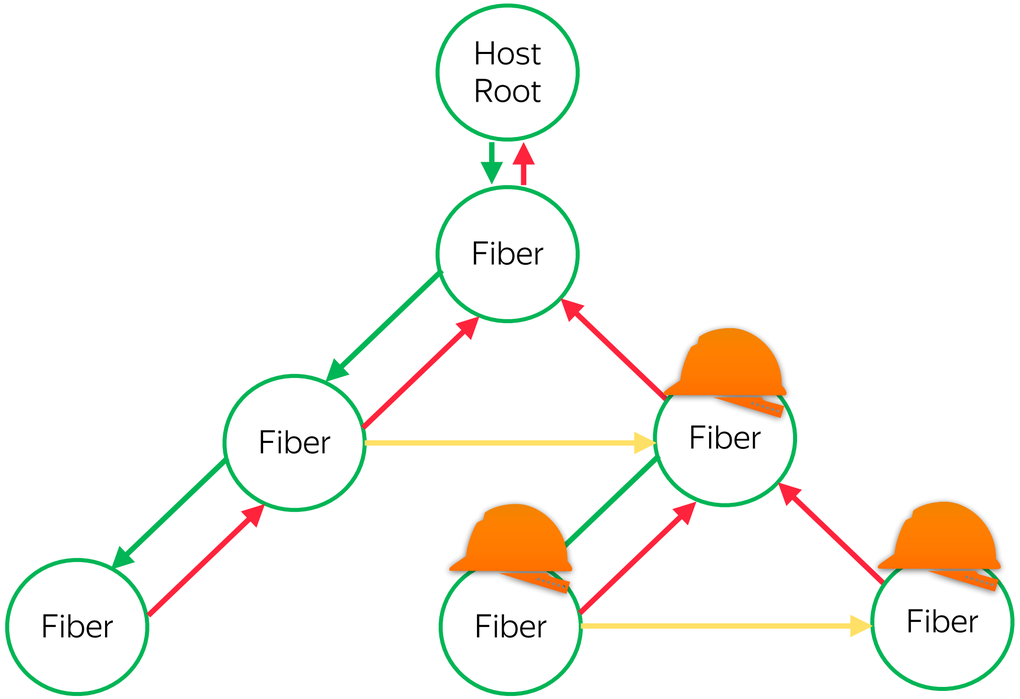


New tree? Yes!

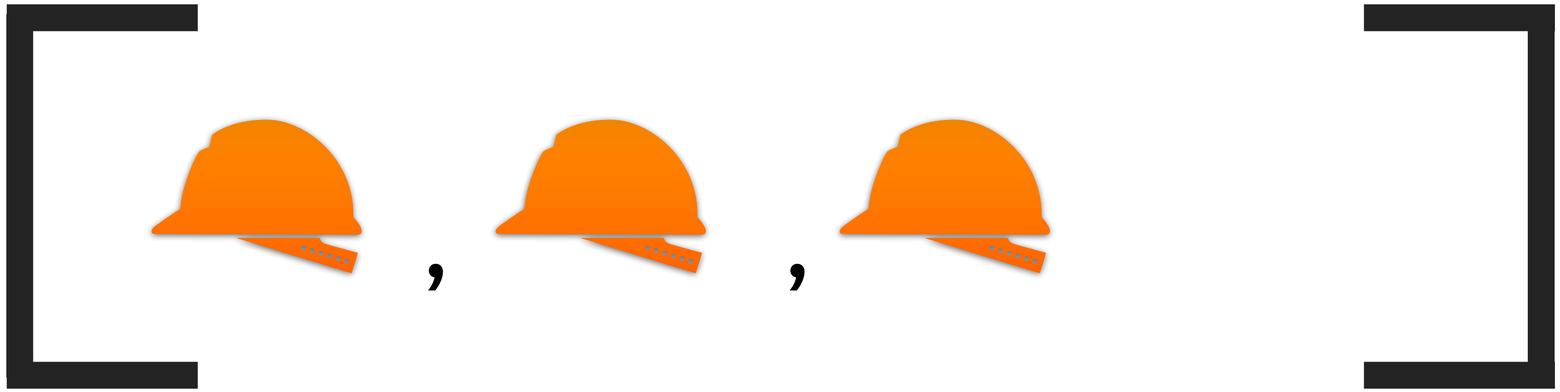
Current tree



WorkinProgress tree



Effect - list!

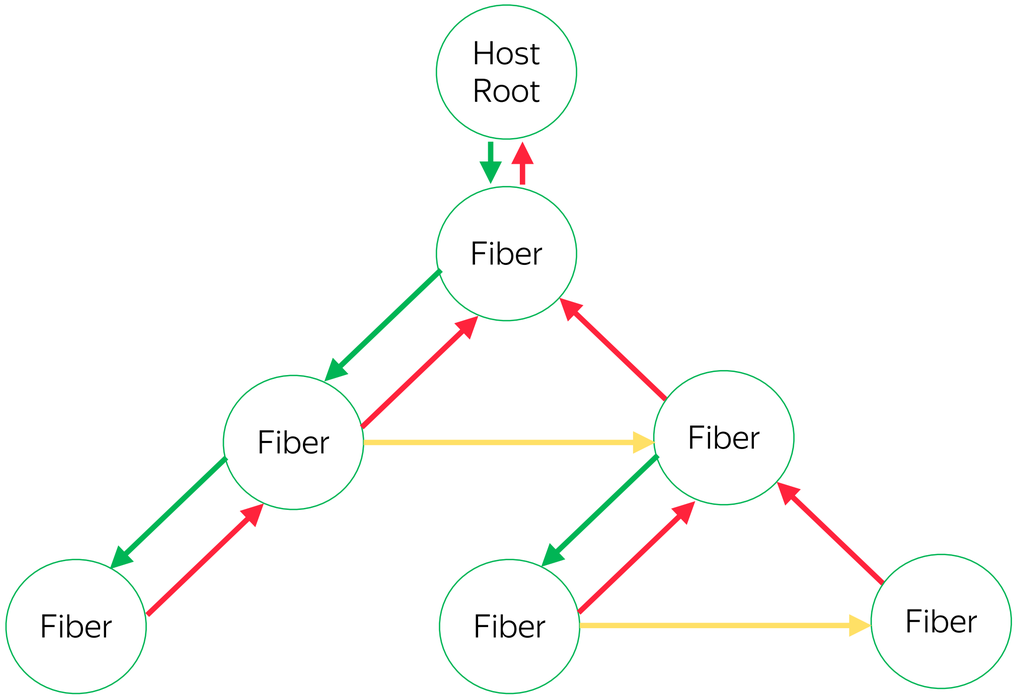


Effect - list!

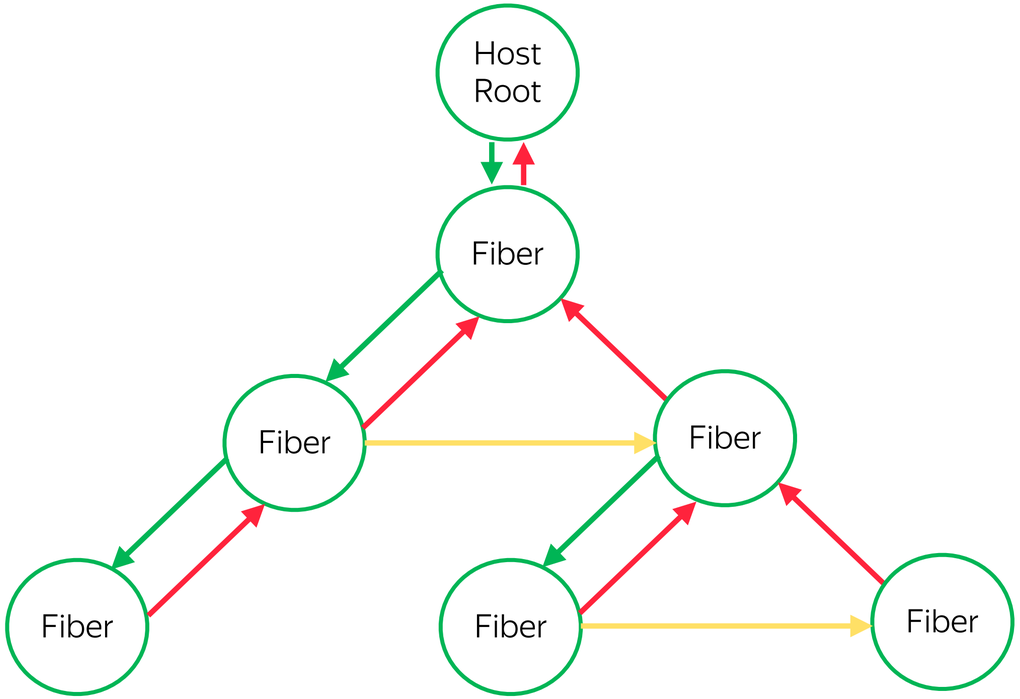
Effect - list!

The King is dead. Long live the King!

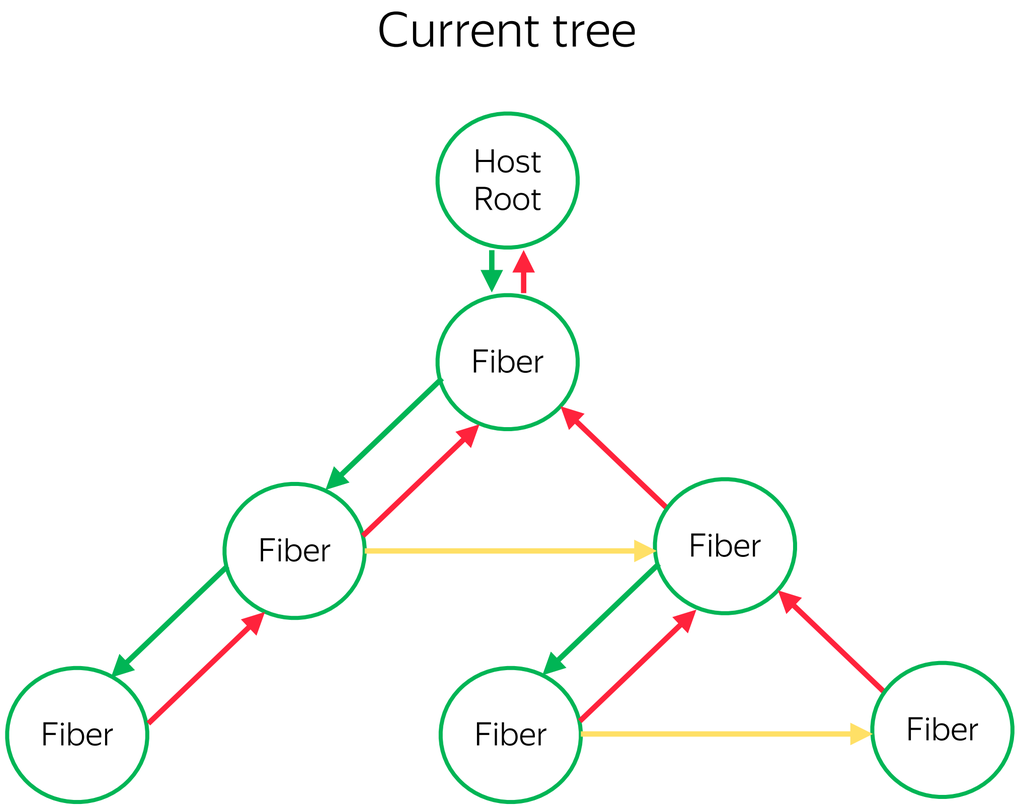
Old Current tree



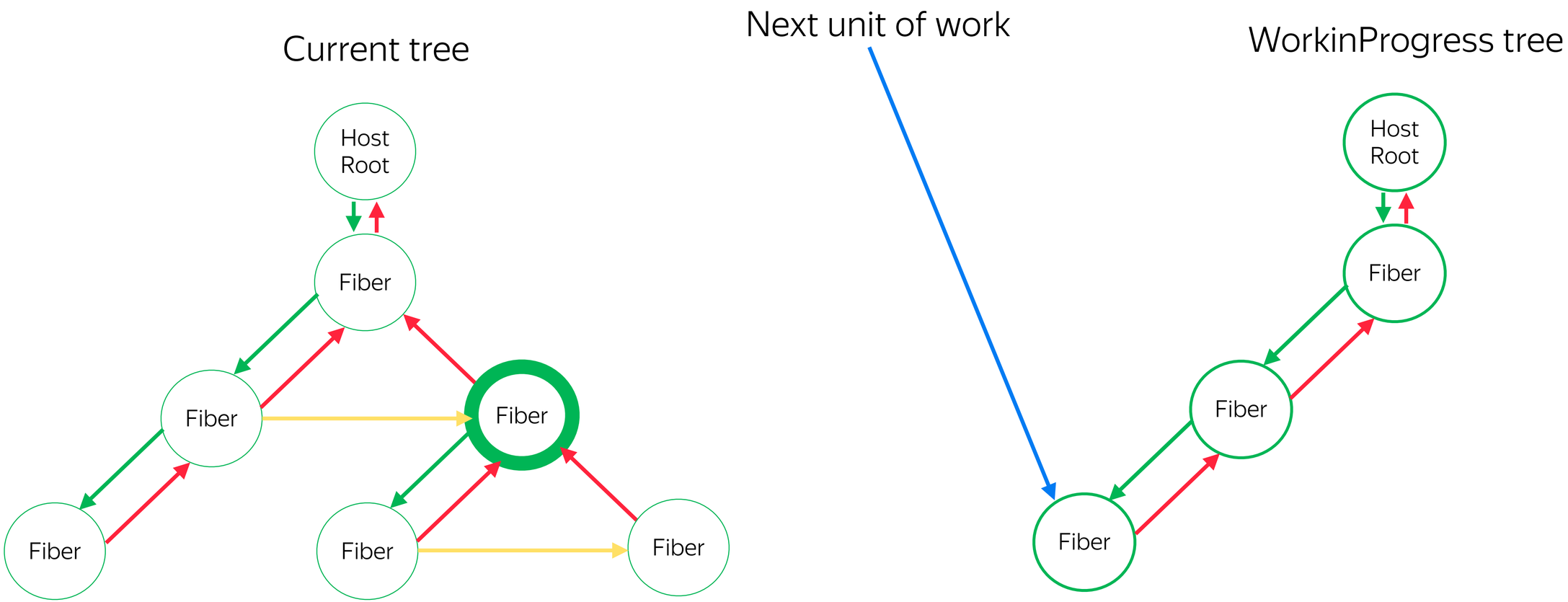
Current tree



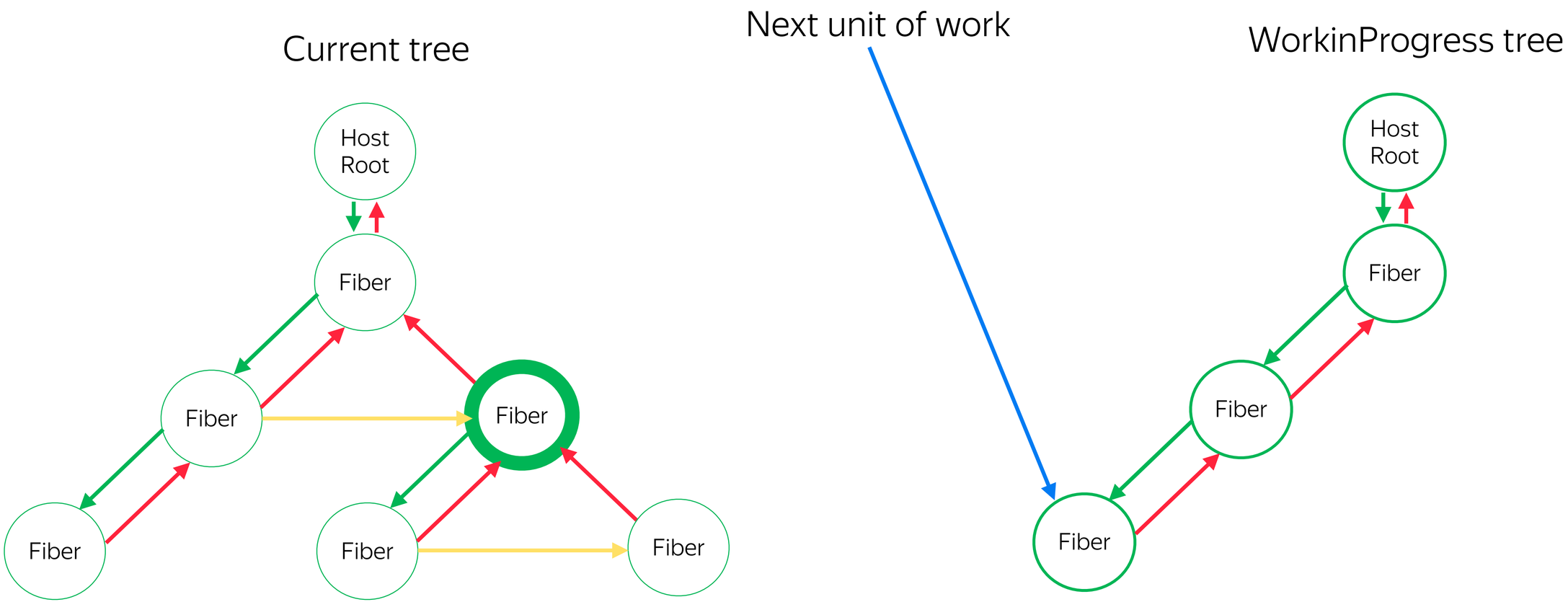
The King is dead. Long live the King!



Этот процесс можно прервать и отменить



Этот процесс можно прервать и отменить



Как React понимает?

Как React понимает?

`requestIdleCallback`

Эвристика

1. **Элементы разных типов - разные деревья;**
2.

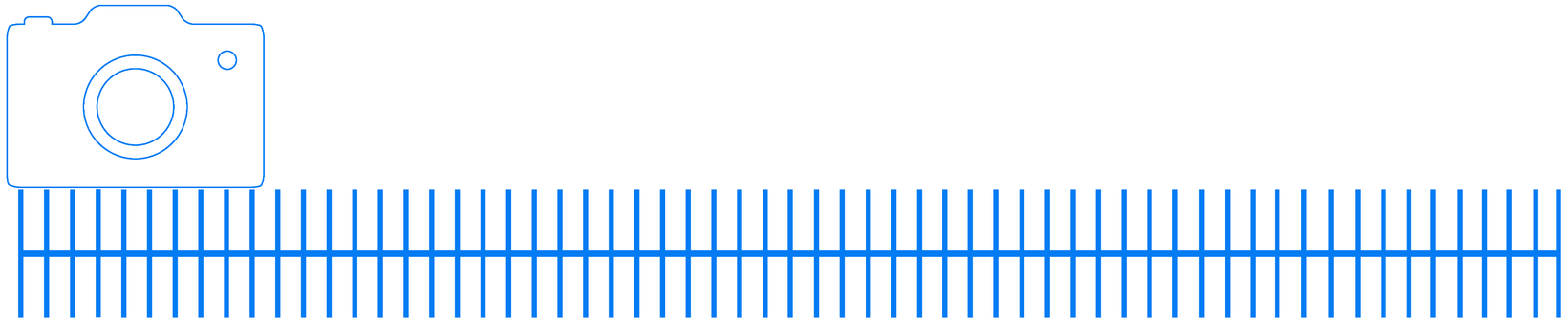
Эвристика

1. Элементы разных типов - разные деревья;

2. **Можно использовать key, чтобы пометить какие элементы будут стабильны в разных рендера**

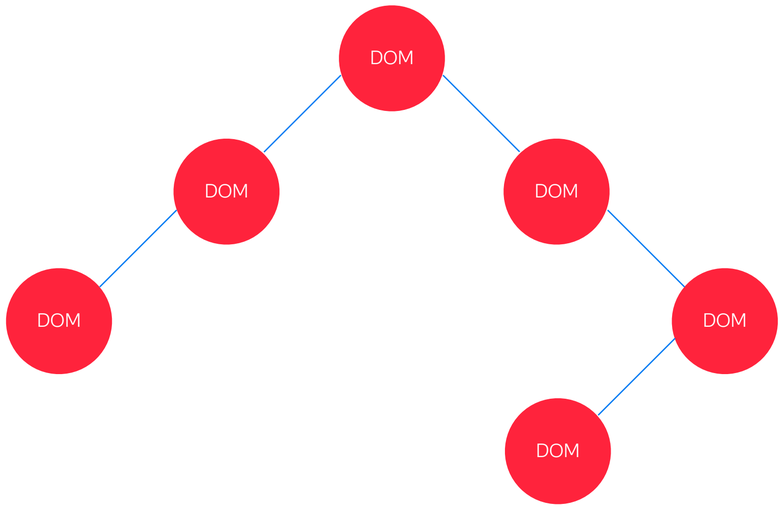
Фаза 2
Commit

60 - кадров в секунду

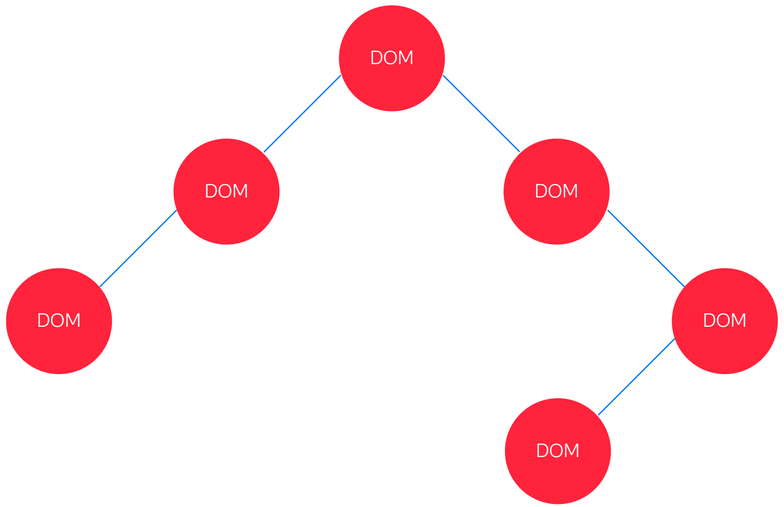


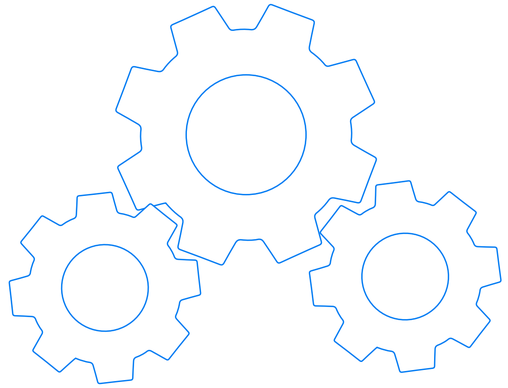
Важно!
Работа выполняется в два подхода

1. Вносим изменения в DOM



2. Запускаем остальные эффекты





Важно!
Фаза не прерывается!!!

Содержание

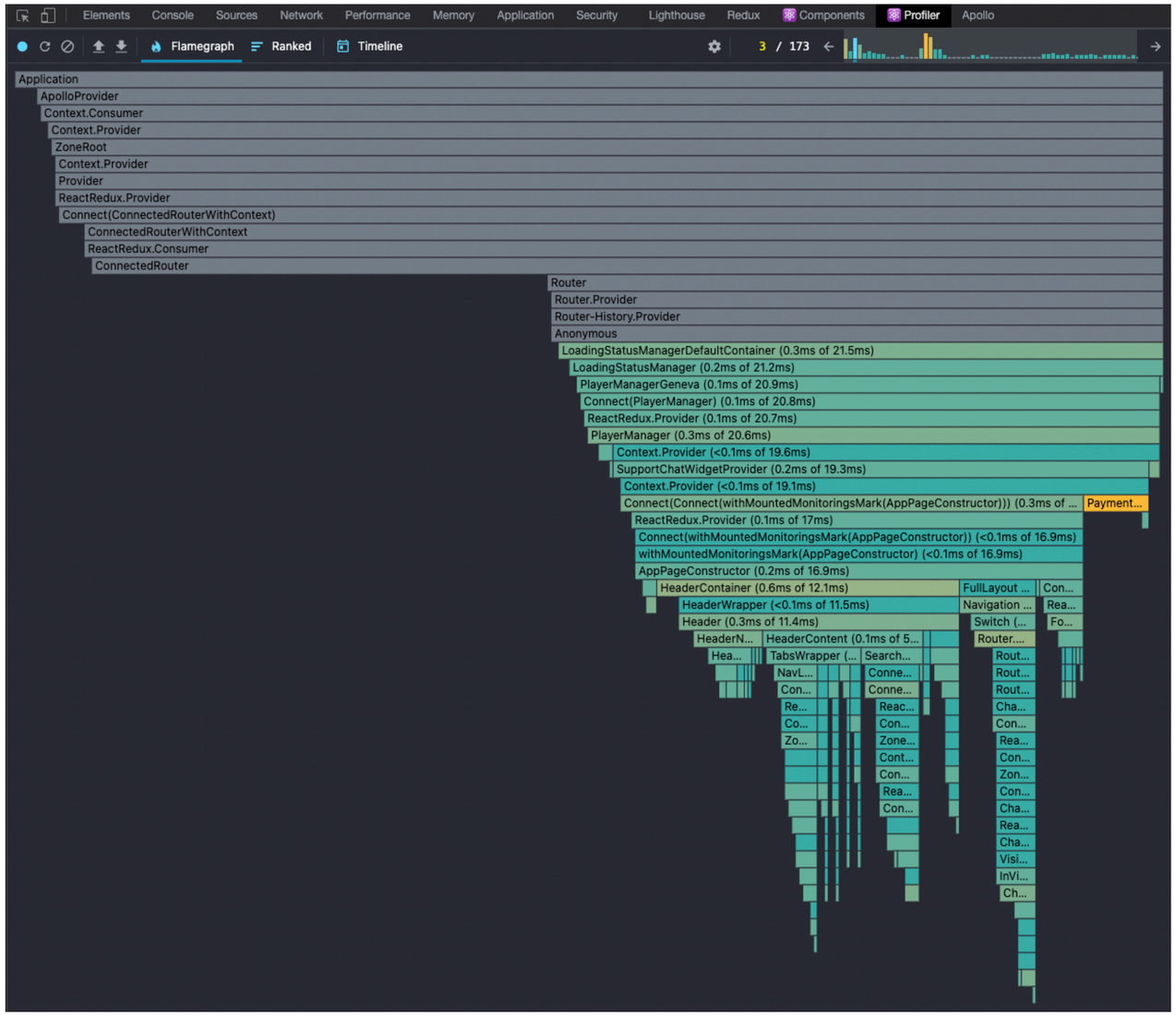
Зачем нам прошлая часть???

Давайте воспользуемся знаниями

```
function Parent() {
    const [count, setCount] = useState(0)

    return <div>
        <span>{count}</span>
        <button onClick={() => setCount(count + 1)}>
          Click me
        </button>
        <ChildVerySlow />
    </div>
}
```

Инструменты



Обнаружен очень медленный компонент

```
function Parent() {
    const [count, setCount] = useState(0)

    return <div>
        <span>{count}</span>
        <button onClick={() => setCount(count + 1)}>
          Click me
        </button>
        <ChildVerySlow />
    </div>
}
```

Обнаружен очень медленный компонент

```
function Parent() {
    const [count, setCount] = useState(0)

    return <div>
        <span>{count}</span>
        <button onClick={() => setCount(count + 1)}>
          Click me
        </button>
        <ChildVerySlow />
    </div>
}
```

Обнаружен источник перерендеров

```
function Parent() {
    const [count, setCount] = useState(0)

    return <div>
        <span>{count}</span>
        <button onClick={() => setCount(count + 1)}>
          Click me
        </button>
        <ChildVerySlow />
    </div>
}
```

Что мы с этим можем сделать?

```
function Parent() {
    const [count, setCount] = useState(0)

    return <div>
        <span>{count}</span>
        <button onClick={() => setCount(count + 1)}>
          Click me
        </button>
        <ChildVerySlow />
    </div>
}
```

Давайте воспользуемся знаниями

```
function Parent() {
    return <div>
        <ChildCount/>
        <ChildVerySlow />
    </div>
}
```

Изолируем источник ререндеров

```
function Parent() {
    return <div>
        <ChildCount/>
        <ChildVerySlow />
    </div>
}
```

State Colocation

```
function ChildCount() {
    const [count, setCount] = useState(0)

    return <>
        <span>{count}</span>
        <button onClick={() => setCount(count + 1)}>
          Click me
        </button>
    </>
}
```

Ремаунты

```
function Parent({ isCountAvailable }) {
    if (isCountAvailable) {
        return <div>
            <ChildCount/>
            <ChildVerySlow />
        </div>
    }

    return <div>
        <ChildVerySlow />
    </div>
}
```

Ремаунты

```
function Parent({ isCountAvailable }) {
    if (isCountAvailable) {
        return <div>
            <ChildCount/>
            <ChildVerySlow />
        </div>
    }

    return <div>
        <ChildVerySlow />
    </div>
}
```

Ремаунты

```
function Parent({ isCountAvailable }) {
    if (isCountAvailable) {
        return <div>
            <ChildCount/>
            <ChildVerySlow />
        </div>
    }

    return <div>
        <ChildVerySlow />
    </div>
}
```

Ремаунты

```
function Parent({ isCountAvailable }) {
    if (isCountAvailable) {
        return <div>
            <ChildCount/>
            <ChildVerySlow />
        </div>
    }

    return <div>
        <ChildVerySlow />
    </div>
}
```

Ремаунты

```
function Parent({ isCountAvailable }) {
    if (isCountAvailable) {
        return <div>
            <ChildCount/>
            <ChildVerySlow key="uniqKey" />
        </div>
    }

    return <div>
        <ChildVerySlow key="uniqKey" />
    </div>
}
```

Ремаунты

```
function Parent({ isCountAvailable }) {
    if (isCountAvailable) {
        return <div>
            <ChildCount/>
            <ChildVerySlow key="uniqKey" />
        </div>
    }

    return <div>
        <ChildVerySlow key="uniqKey" />
    </div>
}
```

HOC

HOC - high order component

HOC - high order component

High order function

Функции высшего порядка — это функции, которые работают с другими функциями, либо принимая их в виде параметров, либо возвращая их.

HOC

```
export const WithAuthorize = (
  isAuthorized,
  { ComponentForAuthorized, ComponentForUnauthorized }
) => {
  const WrappedComponentWithAuthorization = (props) => {
    // check is user authorized
    const isAuthorized = true;

    if (isAuthorized) {
      WrappedComponentWithAuthorization.displayName = `WithAuthorize${ComponentForAuthorized.name}`;
      return <ComponentForAuthorized {...props} />;
    }

    WrappedComponentWithAuthorization.displayName = `WithAuthorize${ComponentForUnauthorized.name}`;
    return <ComponentForUnauthorized {...props} />;
  };

  return WrappedComponentWithAuthorization;
};
```

HOC

```
export const WithAuthorize = (
  isAuthorized,
  { ComponentForAuthorized, ComponentForUnauthorized }
) => {
 //…
};
```

HOC

```
export const WithAuthorize = (
  isAuthorized,
  { ComponentForAuthorized, ComponentForUnauthorized }
) => {
 //…
};
```

HOC

```
export const WithAuthorize = (
  isAuthorized,
  { ComponentForAuthorized, ComponentForUnauthorized }
) => {
 //…
};
```

HOC

```
const WrappedComponentWithAuthorization = (props) => {
  if (isAuthorized) {
    return <ComponentForAuthorized {...props} />;
  }
  return <ComponentForUnauthorized {...props} />;
};

return WrappedComponentWithAuthorization;
```

HOC

```
const WrappedComponentWithAuthorization = (props) => {
  if (isAuthorized) {
    return <ComponentForAuthorized {...props} />;
  }
  return <ComponentForUnauthorized {...props} />;
};

return WrappedComponentWithAuthorization;
```

HOC

```
const WrappedComponentWithAuthorization = (props) => {
  if (isAuthorized) {
    return <ComponentForAuthorized {...props} />;
  }
  return <ComponentForUnauthorized {...props} />;
};

return WrappedComponentWithAuthorization;
```

HOC

```
const FilmDetailsWithAuthorize = WithAuthorize(true, FilmDetails, FilmInfo)
```

HOC

```
<FilmDetailsWithAuthorize/>
```

Memo

```
const ComponentWithMemo = React.memo(Component);
```

Содержание

Pr-r-rops Dr-r-riling

Context

```
export const ThemeContext = React.createContext();
```

Context

```
<ThemeContext.Provider value={value}>
  <RestaurantsPage />
</ThemeContext.Provider>
```

Context

```
const { theme } = useContext(ThemeContext);
```

Portals

Portals

```
<div id="modal-container" />
```

Portals

```
export const Modal = ({children}) => {
    const containerElement = useMemo(
        () => document.getElementById('modal-container'),
        []
    );
    return ReactDOM.createPortal(children, containerElement)
}
```

Portals

```
<Modal><Content /></Modal>
```

Содержание

Создадим простой тогл

```
<Toggle toggleOnText="On" toggleOffText="Off" />
```

Создадим простой тогл

```
<Toggle toggleOnText="On" toggleOffText="Off" />
```

Заказчик хочет еще тогл…

```
<Toggle toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton toggleOnText="On" toggleOffText="Off" />
```

И еще…

```
<Toggle toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButtonAndCustomLabel toggleOnText="On" toggleOffText="Off" />
```

И еще…

```
<Toggle toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButtonAndCustomLabel toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButtonAndCustomText toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton1AndCustomLabel1 toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton2AndCustomLabel toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton3AndCustomLabel2 toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton4AndCustomLabel toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton5AndCustomLabel3 toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton6AndCustomLabel toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton7AndCustomLabel4 toggleOnText="On" toggleOffText="Off" />
```

Compound Components

```
<Toggle toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButtonAndCustomLabel toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButtonAndCustomText toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton1AndCustomLabel1 toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton2AndCustomLabel toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton3AndCustomLabel2 toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton4AndCustomLabel toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton5AndCustomLabel3 toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton6AndCustomLabel toggleOnText="On" toggleOffText="Off" />
<ToggleWithNewButton7AndCustomLabel4 toggleOnText="On" toggleOffText="Off" />
```

Compound Components

```
<Toggle>
  <ToggleOn>The button is on</ToggleOn>
  <ToggleOff>The button is off</ToggleOff>
  <ToggleButton />
</Toggle>
```

Compound Components

```
<Toggle>
  <ToggleOn>The button is on</ToggleOn>
  <ToggleOff>The button is off</ToggleOff>
  <ToggleButton />
</Toggle>
```

Compound Components

```
<Toggle>
  <ToggleOn>The button is on</ToggleOn>
  <ToggleOff>The button is off</ToggleOff>
  <ToggleButton />
</Toggle>
```

Compound Components

```
function Toggle(props) {
  //logic
  return (
    <ToggleContext.Provider value={value}>
      {props.children}
    </ToggleContext.Provider>
  )
}
```

Render props

```
<Layout
  renderToggleOn={(isOn) => <ToggleOn isOn={isOn}/>}
  renderToggleOff={(isOff) => <ToggleOff isOff={isOff}/>}
  renderToggleButton={(isOn) => <ToggleButton isOn={isOn}/>}
/>
```

Render props

```
function Toggle({ renderToggleOn, renderToggleOff, ...otherRenders }) {
      return (
        <div>
          <div className={…}>{renderToggleOn(…)}</div>
          <div className={…}>{renderToggleOff(…)}</div>
           ...
        </div>
      )
    }
```