디지털산업정보학회 논문지 제18권 제1호-2022년 3월
http://dx.doi.org/10.17662/ksdim.2022.18.1.047

사물 웹(WoT) 환경에서 네트워크 모니터링 애플리케이션 개발을 위한 웹 프론트엔드 프레임워크의 적용 현황 및 트렌드

차 시 호*

Status and Trend on Applying Web Front-End Frameworks for Developing Network Monitoring Applications in a Web of Things Environment

Cha Si-Ho

〈Abstract〉

All things connected to the Internet have to ensure interoperability between each other. Web of Things (WoT) is an Internet of Things (IoT) Web standard technology that enables the communication between smart devices and web-based applications. In order for WoT to be possible, monitoring of all devices connected to the Internet have to be possible. To this end, various efforts are being made to develop network monitoring applications using the latest Web front-end frameworks, not traditional web-based monitoring. This paper examines and describes the possibilities of applying Web front-end frameworks such as React, Angular and Svelte to the development of network monitoring applications for WoT. This paper also describes the pros and cons of two major frameworks, React and Angular, in developing monitoring applications that support the cross-platforms and cross-browsers in WoT environments and examines the applicability of them by developing simple network monitoring applications using React.

Key Words : WoT, Web of Things, Network Monitoring, Front-End Frameworks, React

Ⅰ. 서론

사물 인터넷(IoT, Internet of Things)은 인터넷에 연결되는 장치가 더 이상 컴퓨터만이 아니며, 다양한 사물들이 통신 기능을 부여 받은 다양한 센서들을 통하여 인터넷에 연결되어 서로 통신할 수 있는 인터넷 환경을 의미한다[1]. 사물 웹(WoT, Web of Things)은 인터넷에 연결될 수 있는 모든 사물들이 웹과 완전히 통합되는 미래 환경을 설명하는 컴퓨팅 개념이다[2]. 사물 웹 환경 성취하기 위해서는 웹과 통신을 수행하는 스마트 기기들이 기존의 웹 표준을 준수하여 서로 통신할 수 있어야 한다. 이러한 사물 웹은 사물 인터넷의 하위 집합으로 여겨지며, REST, HTTP, URI와 같은 소프트웨어 표준과 프레임워크에 초점을 맞추어 다양한 네트워크 장치들과 결합하고 상호작용하

* 청운대학교 멀티미디어학과 교수

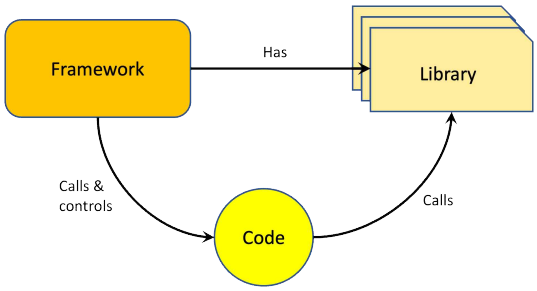
는 애플리케이션 서비스를 만든다. 따라서 사물 웹의 핵심은 기존의 웹 표준을 통신 수단으로 사용하여 웹 서비스에 액세스할 수 있는 일상적인 개체들이라고 생각할 수 있다[2]. 사물 인터넷과 사물 웹의 주요 차이점은 각 기기 간 상호 연결성을 설정하는 계층으로 사물 인터넷은 네트워크 계층만 연결하지만, 사물 웹은 애플리케이션 계층의 연결을 설정한다. 사물 웹은 HTML5와 자바스크립트와 같은 기술적 관점에서 최신 웹의 기술과 표준을 사용하여 웹을 통해 다양한 장치들을 연결할 수 있게 해준다[3].

웹 기반 애플리케이션 시스템을 개발하는데 있어서 기존에는 주로 자바스크립트와 제이쿼리(jQuery) 기반의 프론트엔드(Front-end) 프로그램을 개발하였고, Ajax 통신을 통해 비동기적인 갱신이 가능하도록 구현하였다. 그러나 웹 및 모바일의 발달로 인해 프론트엔드의 개발이 복잡해짐에 따라 필요한 자바스크립트 파일이 많아져 이들을 관리하기가 어려워졌고, 많은 자바스크립트 파일들을 효과적으로 관리해 주기 위한 프론트엔드 프레임워크(Framework)가 등장했다. 프론트엔드 프레임워크는 웹 애플리케이션의 구조에 대한 가이드라인과 기반 코드를 제공하는 개발 도구이다. 최근에는 웹이 복잡해지며 SPA(Single Page Application) 개발을 위한 프레임워크로 각광받고 있다. 현재 가장 많이 사용되는 프론트엔드 프레임워크로는 React[4], Angular[5], Vue[6], Svelte[7] 등이 있다. 이들은 모두 SPA 개발을 위한 프레임워크로 복잡한 웹 애플리케이션 개발에 유용한 웹 프론트엔드 프레임워크이다.

본 논문의 구성은 다음과 같다. 2장에서는 주요 웹 프론트엔드 프레임워크들을 설명하고 이들간의 경쟁과 추세에 대하여 기술한다. 3장에서는 네트워크 모니터링과 관련된 애플리케이션 개발을 위해 웹 프론트엔드 프레임워크가 적용되고 있는 현황과 트렌드에 대하여 기술하고, React를 사용하여 실제 간단한 네트워크 모니터링 애플리케이션을 구현해 본다. 마지막으로, 4장에서는 결론 및 향후 과제에 대하여 기술한다.

II. 관련 연구

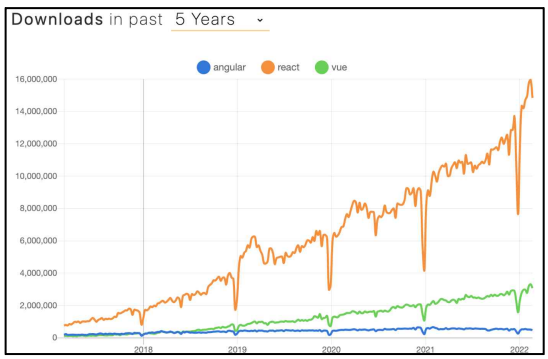
프레임워크란 애플리케이션 구조에 대한 가이드라인과 기반 코드를 제공하는 개발도구이다. 기존의 제이쿼리와 같은 다양한 자바스크립트 라이브러리를 사용하여 프론트엔드 시스템을 개발하는 방식과 달리, 프론트엔드 프레임워크를 사용하면 애플리케이션 구조에 대한 가이드라인과 기반 코드를 제공받을 수 있어서 개발자는 자신이 환경 설정 등에 대하여 신경쓰지 않고 오로지 자신이 개발하고자 하는 코드에만 집중할 수 있다. 라이브러리와 프레임워크의 가장 큰 차이점은 애플리케이션의 제어 흐름에 대한 주도성을 누가 가지는가에 있다. 즉, <그림 1>과 같이 라이브러리는 애플리케이션의 주도성을 사용자 코드가 갖는 반면에 프레임워크에서는 애플리케이션의 주도성을 프레임워크가 갖는다.



<그림 1> 라이브러리와 프레임워크의 차이

현재 가장 많이 사용되고 있는 웹 프론트엔드 프레임워크는 React이다. React[4]는 Facebook에서 개발한 프론트엔트 자바스크립트 프레임워크로 SPA나 모바일 애플리케이션의 개발에 사용된다. Angular[5]는 Google이 만든 오픈소스 자바스크립트 프레임워크로

규모가 큰 애플리케이션을 개발할 때 주로 사용된다. Vue[6]는 에반 유(Evan You)가 커뮤니티와 개발자들의 도움을 받아 완성한 사용자 인터페이스에 특화된 프레임워크이다. Vue는 주로 화면에 보이는 부분에 초점을 맞추기 때문에 다른 프레임워크와 함께 사용할 수 있다. Vue는 React와 같이 가상 DOM (Document Object Model)을 사용하고, 기능적으로 Angular보다 React에 더 가까운 프레임워크이다. Vue는 Angular나 React보다 훨씬 간단하기 때문에 주로 작고 가벼운 애플리케이션을 개발할 때 사용한다. <그림 2>는 React, Angular, Vue에 대하여 어떠한 프레임워크가 가장 많이 다운로딩되어 사용되고 있는지를 나타내는 NPM 트렌드를 보인 것이다. <그림 2>에서 보인 것과 같이 프레임워크들 중에서 React가 인기 있는 이유는 Angular에 비해 쉽고, 크로스 플랫폼 애플리케이션이나 SPA와 같은 현대적인 애플리케이션을 개발할 때 유용하기 때문이다.



<그림 2> NPM 트렌드[8]

Svelte[7]의 가장 큰 특징은 다른 프레임워크들과 달리 컴파일러를 통해 Svelte 코드를 자바스크립트 코드로 바로 변환시킨다는 것이다. Svelte는 다른 프레임워크처럼 애플리케이션을 위한 전체 컴포넌트를 브라우저로 가져오는 대신에 DOM을 갱신하는데 필요한 컴포넌트만 서버의 컴파일러가 자바스크립트 코드로 컴파일함으로써 결과를 빠르게 보여준다. Svelte는 다른 프레임워크에 비해 학습이 용이하고 가볍기 때문에 최근에 가장 사랑받는 프레임워크로 부상하고 있다.

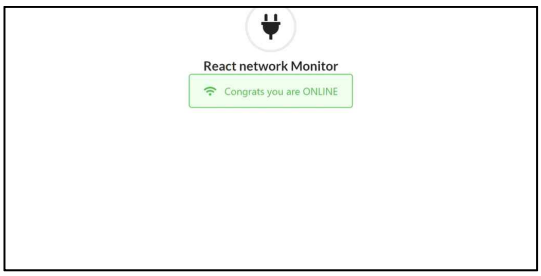
Ⅲ. 웹 프론트엔드 프레임워크의 적용 현황 및 트렌드

3.1 React를 사용한 네트워크 모니터링 애플리케이션

React는 Facebook, Airbnb, Instagram, Netflix, BBC와 같은 글로벌 기업들이 사용하고 있는 가장 인기 있는 웹 프론트엔드 프레임워크이다. React의 특징은 크게 데이터 흐름, 컴포넌트 기반 구조, 가상 DOM, props와 status, JSX(JavaScript XML)로 나눌 수 있다. React의 데이터 흐름은 한 방향으로만 흐르는 단방향 데이터 흐름을 가짐으로써 복잡한 앱에서도 데이터 흐름에서 일어나는 변화를 보다 잘 예측할 수 있다. 부모 컴포넌트로부터 전달되는 데이터를 Props라고 하며, 단방향 데이터 흐름에 따라 이는 자식 컴포넌트에서 변경할 수 없다. 각 컴포넌트는 자신의 State를 가질 수 있으며, 이는 자체적으로 수정이 가능하고, 이 상태 데이터는 자식 컴포넌트에 Props로 전달된다. React에서는 자바스크립트 코드를 기능 단위나 UI 단위의 컴포넌트로 구현한다. 이를 통해 여러 개의 독립된 컴포넌트를 조립하여 하나의 기능 및 화면을 구성함으로써 애플리케이션이 복잡해지더라도 코드의 유지보수 및 관리가 용이하다. 또한 JSX를 사용함으로써 자바스크립트 코드 내에서 다양한 HTML 요소들을 보다 시각적으로 생성할 수 있다.

Deven Rathod[9]는 React의 함수형 프로그래밍인 Hook을 사용하여 간단한 네트워크 모니터링 애플리

케이션을 구현함으로써 사물 웹 환경에서의 React의 활용 가능성을 보여주었다. Deven의 애플리케이션은 React 애플리케이션을 사용하는 동안 네트워크의 연결이 끊기는지를 확인할 수 있는 사용자 네트워크 연결 상태를 표시해준다. 이 애플리케이션이 작동하기 위해서는 브라우저가 네트워크 연결을 끊거나 다시 연결될 때 실행되는 코드를 작성해야 한다. 이를 위해 src 폴더에 onlinestatus 컴포넌트(onlinestatus.js)를 React hook을 작성하여, 브라우저의 온라인/오프라인 이벤트에 대한 리스너를 등록하고, 해당 이벤트가 발생하면 온라인 값을 true 또는 false로 설정할 수 있다. App 컴포넌트에서는 이 hook를 컴포넌트로 임포트하고, 이 hook가 State 변수인 online으로 네트워크의 연결 상태를 관리할 수 있다. <그림 3>은 Deven의 애플리케이션 실행화면을 캡처한 것으로 온라인 값인 true를 State 변수가 받아 해당 내용을 하면에 보여준 것이다. Deven은 이와 같이 React를 사용하여 간단한 네트워크 모니터링 애플리케이션을 만들 수 있음을 보였다. 이 프로젝트는 추가 코드를 작성하여 서버 및 그 이상으로의 연결을 확인하도록 애플리케이션을 확장할 수 있다.



<그림 3> Deven의 애플리케이션 실행 결과[9]

Thomas Sentre[10]는 Deven이 작성한 애플리케이션[9]을 보완 확장하였다. Thomas는 src 폴더 안에 component 폴더를 만들고, 네트워크의 연결 여부를 알려주는 React hook인 useOnline 컴포넌트(useOnline.js)를 생성하였다. 이 hook은 Deven의 onlinestatus hook과 기능이 동일하다. Deven의 프로젝트와 같이 useOnline hook는 App 컴포넌트에서 호출되고, online State를 통해 네트워크의 연결 상태를 관리하였다. <그림 4>는 useOnline 컴포넌트의 자바스크립트 코드를 보인 것이고, <그림 5>는 App 컴포넌트의 자바스크립트 코드를 보인 것이다.

src > components > JS useOnline.js > ...

```
1   import  {useEffect, useState } from 'react';
2   const useOnline=()=>{
3       const [online,setOnline]=useState(navigator.onLine);
4       useEffect(()=>{
5           if(window.addEventListener){
6               window.addEventListener('online',
7               ()=>setOnline(true),false);
8               window.addEventListener('offline',
9               ()=>setOnline(false),false);
10          }else{
11              document.body.ononline=()=>setOnline(true);
12              document.body.onoffline=()=>setOnline(false);
13          }
14      },[])
15          return online;
16      };
17  export default useOnline;
```

<그림 4> useOnline 컴포넌트(useOnline.js)[10]

```
1   import 'App.css';
2   import useOnline from './components/useOnline.js';
3   function App() {
4     const online=useOnline();
5     return (
6       <div className="App">
7         <h3>Network connection checker</h3>
8         <div className="connection-status">
9         {
10          online ?
11          (
12            <div className="indicator-online" style={{color:"green",fontSize:"2rem"}}>
13              You are now ONLINE
14            </div>
15          ) :
16          (
17            <img src="https://i.ibb.co/JB8MmZs/network-status.png" alt="network status"/>
18          )
19        }
20        </div>
21      </div>
22    );
23  }
24  export default App;
```

<그림 5> App 컴포넌트(App.js)[10]

Thomas[10]와 Deven[9]은 React 프레임워크를 사용하여 같은 기능을 수행하는 애플리케이션을 개발하였으므로 프레임워크의 가이드라인 특성 상 거의 유사한 자바스크립트 코드를 작성하였다.

Thomas의 애플리케이션을 실행하면 실행 결과 페이지에 네트워크의 상태가 온라인 상태로 표시된다.

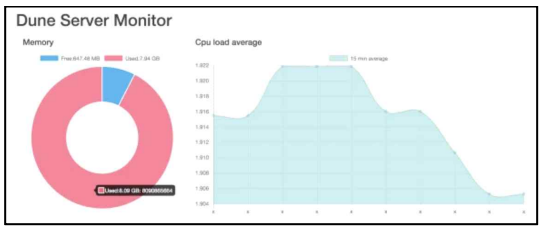
이 때 네트워크 연결을 끊으면 <그림 6>과 같은 페이지가 자동으로 다시 렌더링된다. 이는 React는 특성상 state 값이 바뀌면 자동으로 렌더링을 다시 시작하기 때문이다.

<그림 6> Thomas의 애플리케이션 실행 결과[10]

3.2 Angular를 사용한 서버 모니터링 애플리케이션

Angular는 프론트엔드 웹 애플리케이션 개발을 위해 구글에서 구현한 오픈 소스 프레임워크이다. Angular는 웹, 모바일 웹, 네이티브 애플리케이션, 네이티브 데스크톱까지 모든 플랫폼에서 재사용할 수 있는 애플리케이션을 개발할 수 있도록 지원한다. Angular는 기본적으로 SPA 개발을 위한 프레임워크이지만 서버 사이트 렌더링을 위한 기능도 제공해주며, 파일의 생성부터 빌드와 패키징 및 서버 기능까지 개발에 필요한 거의 모든 기능을 자체적으로 제공한다.

Deven Rathod는 풀스택(full stack) 개발자로서 앞 절에서 기술한 React를 활용한 네트워크 모니터링뿐만 아니라, Angular를 사용하여 서버를 모니터링하는 애플리케이션을 개발하였다[11]. Deven의 이 프로젝트는 우리의 서버 상에서 사용가능한 메모리 용량을 보여주기 위해 Node[12]와 Angular를 사용하여 실시간 서버 모니터링 애플리케이션을 구현하였다. 이 프로젝트는 Node와 Angular 이외에도 서버 개발을 위해 Vagrant[13]를 사용하였으며, 모니터링 결과를 차트의 형태로 나타내기 위해 char.js[14]을 사용하였다. <그림 7>은 Deven이 개발한 서버 모니터링 실행 결과를 캡처한 것으로, 해당 서버의 OS는 Node가 인스톨된 Ubuntu 16.04로 Vagrant를 사용하여 구축된 서버에 대한 모니터링 결과를 보인 것이다.



<그림 7> Angular 기반 서버 모니터링 애플리케이션의 실행 결과[11]

3.3 웹 프레임워크의 적용 트렌드

사물 웹 환경을 위한 웹 애플리케이션이나 모바일 앱을 개발하기 위해서는 React나 Angular 중 어떤 프레임워크를 선택해야 하는가의 문제에 직면하게 된다. 따라서 개발자의 관점에서 각자의 프로젝트에 합당한 프레임워크를 선택할 수 있도록 이들의 장단점을 정리하였다.

<표 1> React의 장점과 단점

장점	단점
저렴한 개발 비용	빈번한 버전 업데이트
개발자들에게 많은 인기	불완전성
우수한 성능	
개발 기간 단축	
뛰어난 사용자 경험	
테스팅과 디버깅 용이	

<표 1>은 네트워크나 서버의 모니터링을 위한 애

플리케이션의 개발자 관점에서 React의 장점과 단점을 정리한 것이다. <표 1>에서 보인 것과 같이 React는 많은 장점을 제공하지만, React는 여전히 새로운 기술이며 빠르게 새로운 버전으로 업데이트되고 있기 때문에 개발자들이 이 속도를 유지하기 어려울 수 있다. 또한 다양한 기능이 필요한 경우 다른 기술에 의존해야 하기 때문에 모든 프로젝트가 다르게 보일 수도 있다. <표 2>는 앞에서 설명한 React에 비하여 Angular의 장점과 단점을 정리한 것이다.

<표 2> Angular의 장점과 단점

장점	단점
오류 처리 더 높은 성능 Angular CLI를 통한 원활한 업데이트 완전한 MVC 프레임워크	방대한 규모 학습해야 할 내용이 많음

React는 자바스크립트를 개발 언어로 사용하며 필요시 타입스크립트를 활용하지만, Angular는 개발 언어로 타입스크립트를 사용한다. 호환성에 있어서 React는 완전한 이전 버전과의 호환성을 제공하지만 Angular는 업데이트가 요구된다. 또한 React는 가상 DOM을 사용하고, Angular는 리얼 DOM을 사용하는 차이점이 존재한다.

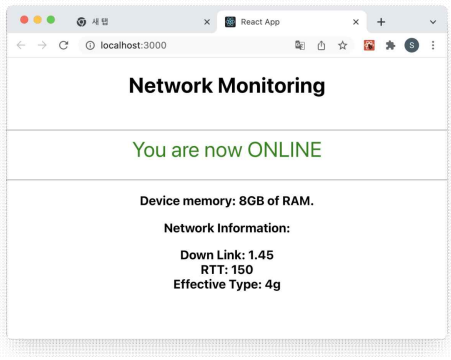
3.4 React로 간단한 네트워크 모니터링 애플리케이션 구현하기

본 절에서는 3.1절에서 기술한 Thomas[10]가 구현한 React 기반의 네트워크 모니터링 애플리케이션을 보다 많은 정보를 모니터링할 수 있도록 업데이트하고자 한다. 여기서 React를 사용한 이유는 Angular보다 React가 구현하기 쉽고, 단순하며, 이미 우리에게 익숙한 자바스크립트를 사용하여 개발할 수 있기 때문이다. 여기서 구현한 애플리케이션은 기존의 Thomas가 구현한 애플리케이션의 업데이트 버전이라 할 수 있다. 우리의 애플리케이션에서는 서버의 메모리(RAM) 정보를 모니터링하는 컴포넌트인 useDeviceMemory.js와 다양한 네트워크 정보를 모니터링하는 컴포넌트인 NetworkInformation.js를 components 디렉토리에 추가로 구현하였다.

<그림 8>은 본 논문에서 업데이트한 App 컴포넌트를 보인 것이다. App 컴포넌트는 서버의 메모리 정보와 네트워크 정보를 모니터링하기 위해

```
src > JS App.js > ...
1   import './App.css';
2   import useDeviceMemory from './components/useDeviceMemory';
3   import NetworkInformation from './components/NetworkInformation';
4   import useOnline from './components/useOnline';
5
6   function App() {
7     const online = useOnline();
8     const memory = useDeviceMemory();
9
10    return (
11      <div className="App">
12        <h1> Network Monitoring</h1>
13        <br />
14        <hr />
15        <div className='connection-status'>
16          {
17            online ?
18            (
19              <div className='indicator-online' style={{color:"green", fontSize:"2rem"}}>
20                You are now ONLINE
21              </div>
22            ) :
23            (
24              <img src="../public/network-disconnection.png" alt="network status" />
25            )
26          }
27        </div>
28        <br />
29        <hr />
30        <div>
31          <h3>Device memory: {memory}GB of RAM.</h3>
32          <h3>Network Information:</h3>
33            <NetworkInformation />
34        </div>
35      </div>
36    );
37  }
38
39  export default App;
```

<그림 8> App 컴포넌트(App.js) 구현 코드



<그림 9> 네트워크 모니터링 실행 결과

useDeviceMemory와 NetworkInformation 컴포넌트를 사용하였다. 우리의 구현에서 useDeviceMemeory는 hook으로 구현하였고, NetworkInformation은 일반 컴포넌트로 개발하였다.

<그림 9>는 이렇게 개발된 애플리케이션을 실행한 결과를 캡처한 것이다. 이 결과에서 알 수 있듯이 React는 기능별로 혹은 UI별로 컴포넌트를 개발함으로써 쉽게 우리가 원하는 크로스 플랫폼 애플리케이션을 사물 웹 환경에서 개발할 수 있다.

Ⅳ. 결론

본 논문에는 모든 사물들이 웹에 연결되는 사물 웹 환경을 위하여 최신의 웹 표준 기술을 사용하여 사물들의 상태를 모니터링할 수 있는 애플리케이션의 개발에 대한 현황과 트렌드에 대하여 기술하였다. 특히 웹 프론트엔드 프레임워크로 경쟁하고 있는 React와 Angular을 활용한 개발 예를 살펴보고, 이들 프레임워크의 장단점도 살펴보았다. 또한 React 기반의 간단한 네트워크 모니터링 애플리케이션을 구현해 봄으로써 크로스 플랫폼 및 크로스 브라우저를 지원하는 웹 애플리케이션을 웹 프론트엔드 프레임워크로 개발할 수 있음을 보였다. 향후 과제로, 본 논문에서 구현한 React 기반의 네트워크 모니터링 애플리케이션이 실제로 유용하게 사용되기 위해서는 많은 컴포넌트들이 추가로 개발되어야 하고, 웹에 연결되는 모든 사물들을 관리할 수 있도록 확장성을 확보해야 할 것이다.

참고문헌

[1] L. K. Ramasamy, S. Kadry, “Internet of things (IoT)”, Blockchain in the Industrial Internet of Things, IOP Publishing Ltd. 2021, pp.1-16.

[2] Techopedia, Web of Things (WoT), https://www.techopedia.com/definition/26834/web-of-things-wot.

[3] MobiDev, Introduction to the Web of Things, Dec. 16, 2021, https://www.iotforall.com/introduction-to-the-web-of-things.

[4] React, https://reactjs.org/.

[5] Angular, https://angular.io/.

[6] Vue, https://vuejs.org/.

[7] Svelte, https://svelte.dev/.

[8] NPM trends, https://www.npmtrends.com/angular-vs-react-vs-vue

[9] D. Rathod, Build a Network Monitoring Application using React, Mar. 07, 2021, https://codesource.io/build-a-network-monitoring-app-using-react/.

[10] T. Sentre, Build a Network Monitoring Application using React, Nov. 22, 2021, https://medium.com/@merndev/build-a-network-monitoring-application-using-react-7bd9170b5f5a.

[11] D. Rathod, Build a server monitoring app using Node & Angular, Dec. 17, 2019, https://codesource.io/build-a-server-monitoring-app-using-node-angular/.

[12] Node, https://nodejs.org/en/.

[13] Vagrant, https://www.vagrantup.com/.

[14] chart.js, https://www.chartjs.org/.

■ 저자소개 ■

차 시 호
Cha, Si-Ho

2009년 3월~현재
청운대학교 멀티미디어학과 교수
2020년 3월~2021년 2월
Auckland University of Technology 방문교수
2004년 2월 광운대학교 컴퓨터과학과(공학박사)
1997년 7월~2000년 2월 대우통신 종합연구소 선임연구원

관심분야 : 네트워크 관리, 차량통신 네트워크, 시맨틱웹, 머신러닝
E-mail : shcha@chungwoon.ac.kr

논문접수일 : 2022년 3월 7일
수 정 일 : 2022년 3월 16일
게재확정일 : 2022년 3월 19일