Вісник Тернопільського національного технічного університету
https://doi.org/10.33108/visnyk_tntu
Scientific Journal of the Ternopil National Technical University
2021, № 4 (104) https://doi.org/10.33108/visnyk_tntu2021.04
ISSN 2522-4433. Web: ***visnyk.tntu.edu.ua***

UDC 004.772

TRAFFIC OPTIMIZATION IN WIFI NETWORKS FOR THE INTERNET OF THINGS

Vyacheslav Starchenko

Petro Mohyla Black Sea National University, Mykolayiv, Ukraine

***Summary**. One of the main problems of modern IoT networks is the large amount of automated traffic generated by their nodes. This puts a significant strain on modern communications networks, which will only increase over time. One way to overcome this problem is to optimize the data structure and improve the methods of data collection, transmission and processing. The purpose of this study is to optimize traffic in the IoT network at the level of software architecture and data representation. The object of the study is the FireBeetle Covers-24 x 8 LED Matrix ESP32, which is controlled by the HOLTEK HT1632C controller, connected via WiFi – interface with the IoT network, implemented on the basis of the ESP8266 microcontroller. The subject of the research is the process of optimizing program code by choosing the most efficient software architecture. The article considers the three most common software architectures and gives examples of their implementation. The architecture based on REST technology was the first to be considered. This technology is the most popular and widespread due to the simplicity of concept and implementation. But its disadvantage is the significant congestion of the communication line with a large amount of the same type of service information. Reducing the amount of service information by optimizing the HTML page code using JavaScript is demonstrated by the example of the second software architecture. But such optimization does not allow to completely separate the static and dynamic components of the information transmitted by the communication line. This division can easily be done within the software architecture based on Ajax & JSON, an example of which is given in the third. The great advantage of this architecture is that the static component is transmitted by the network only once at the beginning of the communication session. Then only the dynamic component is transmitted. According to the results of testing the developed hardware and software module and comparing the amount of generated data transmitted by WiFi network, it is shown that the software architecture based on Ajax & JSON has the highest network efficiency, significantly reducing network traf6fic compared to others.*

***Key words:** wireless technology, software architecture, Internet of Things, WiFi, JavaScript, Ajax, REST, JSON.*

https://doi.org/10.33108/visnyk_tntu2021.04.131 *Received 20.11.2021*

Introduction. Along with Neural Networks, Artificial Intelligence and Machine Learning, the Internet of Things or IoT is another rapidly developing technology. From the point of view of architecture, IoT is a network of networks consisting of uniquely identified objects – «things» which can interact with each other via IP connections without human intervention [1]. This creates a fundamentally new phenomenon – the industrial Internet.

The practice of collecting and analyzing data about an object whether it is a mechanism, a building or a person has existed for a long time due to sensors. The industrial Internet is radically different as the sensors are integrated into a single network with analytics and/or control systems. Thus, the industrial object forms its own independent network, where there is an intensive exchange of data, on the basis of which decisions are automatically made and actions are taken to control this object. Elements of artificial intelligence and principles of self-regulation are widely used here. This adds digital intelligence to the devices which would otherwise be inactive, allowing them to communicate without human intervention, and creates opportunities to merge digital and physical worlds.

Due to microprocessors and wireless networks, it is possible to convert everything from tablet to airplane into IoT. It is assumed that in the future Internet the things will become active

Corresponding author: Vyacheslav Starchenko; e-mail: viacheslav.starchenko@chmnu.edu.ua

participants in business, information and social processes, where they can interact with each other, exchanging information about the environment without human intervention.

At present, the concept of IoT has been already referred to the billions of physical devices around the world connected to the Internet, and constantly collect, transmit and analyze large amounts of data. This results in significant load on modern communication networks, which will only increase over time.

There are two ways of solving this problem. The first one is to increase the capacity of communication networks by developing new standards and data transmission technologies. So 5G wireless technology is developed taking into account the requirements of IoT [2]. The second one is to optimize the structure of the data itself and improvement the methods of their collection, transmission and processing. Such optimization is possible at every IoT level. Unfortunately, the IoT standard reference model has not yet been adopted. The most authoritative are the standard model of IoT World Forum [3], and Overview of the Internet of Things Recommendation ITU-T Y.2060 [4], published by the International Telecommunication Union. Therefore, there is no single strategy and methodology for such optimization.

The objective of the paper is to investigate the possibilities for traffic optimization in the IoT network at the level of software architecture and data representation.

Hardware component of the system. The FireBeetle Covers-24 x 8 LED Matrix [5] consisting of 192 LEDs arranged in 8 rows having 24 elements in each is selected as the control object [6]. The matrix is controlled by HOLTEK HT1632C controller [7]. HT1632C is controlled by serial interface via DATA, WR and RD contacts. Power supply is 3.3V through VCC and GND contacts. The controller has integrated RAM with individual addressing of each LED and 256 kHz RC oscillator which makes it possible to control LEDs by pulse-width modulation. So each LED can be controlled separately with 16-level brightness adjustment. The relative position of LEDs and control contacts of the matrix is shown in Figure 1.

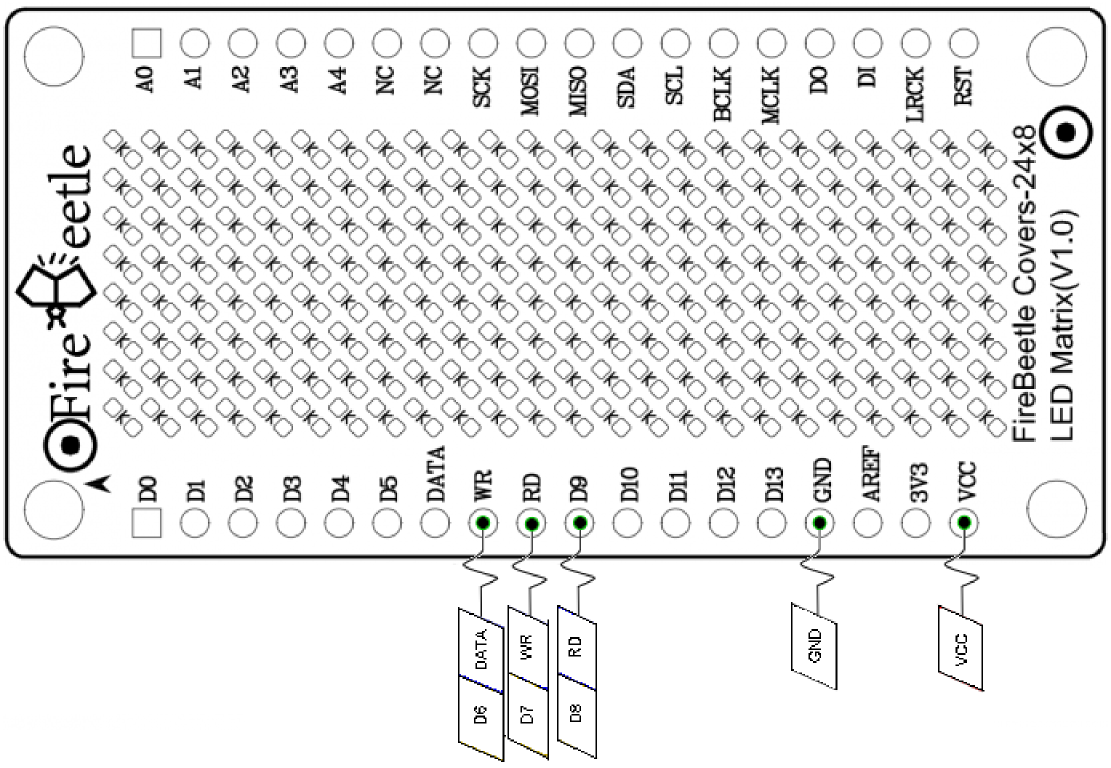


Figure 1. Scheme of FireBeetle Covers-24 x 8 LED Matrix control contacts

The interface with WiFi network is provided by ESP8266 microcontroller [8]. The controller supports WiFi standards IEEE802.11 b/g/n within the range from 2.4 GHz to 2.5 GHz, has 32-bit processor running at 80MHz, SRAM – 50KB, and Flash – 16 Mb. ESP8266 is powered by 3.3V voltage. The general hardware architecture of the system is shown in Figure 2.

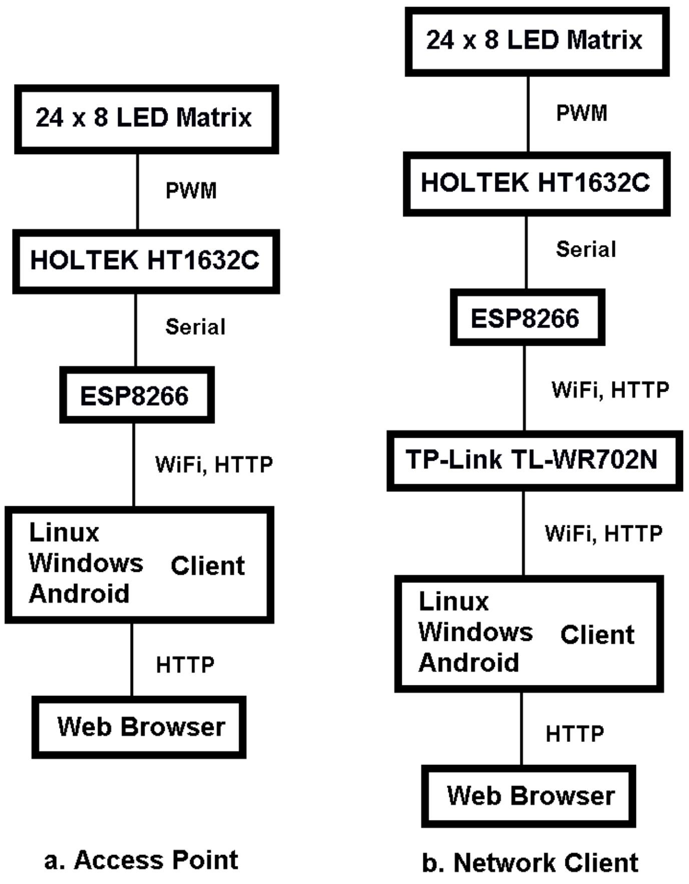


Figure 2. The overall hardware architecture of the system (a) in access point mode, (b) in client mode

The system can operate in both WiFi access point mode and WiFi client mode. In access point mode, the system provides all the necessary stack of protocols required for the operation of WiFi network. In client mode, the system can be integrated into any compatible WiFi network with IEEE802.11 b/g/n protocols by connecting to its local router. The TP-Link TL-WR702N router is used in this investigation [9].

Software component of the system. As a rule, the software architecture of such systems is based on the use of REST technology [10, 11]. It has significant advantages over competitors such as SOAP [12] and WSDL [13, 14]. The main advantages are as follows:

1. simplicity and transparency of software interfaces;
2. program model focused on resources;
3. prevalence of technology;
4. possibility of JavaScript optimization;
5. easy integration with Web 2.0 technology.

This allowed us to choose REST technology as the basis for the investigation.

Implementation of REST concept. For effective system control, the customer should receive complete information about the status of LED module. To do this, the server generates a WEB page of the LED module, which according to REST concept contains descriptions of the states of all LED elements, as shown in Listing 1.

```
int display[8][24];

void handleRoot() {
  String message = "<html><body>";
  message += "      <table>";
  for (int r = 0; r < 8; ++r) {
    message += "<tr>";
    for (int c = 0; c < 24; ++c) {
      message += "<td class=";
      message += ((display[r][c] == 1)? "on" : "off");
      message += "><a href=/";
      message += ((display[r][c] == 0)? "on" : "off");
      message += "</a></td>";
      }
    message += "</tr>";
    }
  message += "      </table>";
  message += "  </body>";
  message += "</html>";
  server.send(200, "text/html", message);
}
```

Listing 1. Function of generating the title page by Web server of LED module according to REST concept

In the above given listing, the handleRoot function generates HTML table, each cell of which represents the state of the corresponding LED. All HTML code is generated on the side of the module by its server and transmitted by communication line to the client's browser. This significantly overloads the communication line because it transmits a lot of the same type of service information required to build the cells of HTML table itself.

JavaScript optimization. To optimize network traffic, all the information generated by the handleRoot function must be divided into three components:

- information required to build HTML page itself;
- information required to build LED matrix state HTML table;
- information about the status of each LED matrix.

The first and second components are static. The third component presents the dynamics of changes in the states of the LEDs of the matrix.

At the first stage of optimization, network traffic can be significantly reduced by placing the task of generating the table on the client's browser. This can be easily done due to the optimization of HTML page code by JavaScript language. The example of such optimization is shown in Listing 2.

In this listing, the handleRoot function, along with the HTML code, according to the REST concept, generates a JavaScript array containing the states of each LED of the LED matrix, and JavaScript code to convert the data of this array into an HTML table. Thus, the optimization is carried out by reducing the amount of service information required to build

HTML state table of the LED matrix. Only information required to build HTML page and information about the status of LED matrix is transmitted from the server to the client. But the amount of service information required to build HTML table is always proportional to its size.

```
     int display[8][24];

     void handleRoot() {
       String message = "<html>";
       message += "  <head><script>";
       message += "      var d = [";
       for (uint8_t row = 0; row < 8; ++row){
         message += "[ ";
         for (uint8_t col = 0; col < 24; ++col){
           message += display[row][col];
           message += ",";
         }
         message += "],";
       }
       message += "        ];";
       message += "    </script></head><body><script>";
       message += "      var s = \"<table>\";";
       message += "      for (var r = 0; r < 8; ++r) {";
       message += "        s += \"<tr>\";";
       message += "        for (var c = 0; c < 24; ++c) {";
       message += "          s += \"<td class=\" +
         ((d[r][c] == 1)? \"on\" : \"off\") + \"><a href=/\" +
         ((d[r][c]  ==  0)?  \"on\"  :  \"off\")  +  \"?x=\"+  c  +
\"&y=\" +
         r + \">\" + c + \" / \" + r + \"</a></td>\";";
       message += "          }";
       message += "        s += \"</tr>\";";
       message += "        }";
       message += "      s += \"</table>\";";
       message += "      document.write( s);";
       message += "    </script></body></html>";
       server.send(200, "text/html", message);
     }
```

Listing 2. JavaScript optimization of the cover page generation function by Web server of the LED module

Ajax & JSON optimization. At the second stage of optimization, significant reduction in network traffic can be achieved by reducing the static component of information generated by Web server. This can be implemented by means of Ajax technology [15, 16, 17] using JSON data presentation format [18, 19, 20].

```
void handleJsonState() {
  String message = "[ ";
  for (uint8_t row = 0; row < 8; ++row){
    message += "[ ";
    for (uint8_t col = 0; col < 24; ++col){
      message += display[row][col];
      message += (col < 23)? ", ": " ";
    }
    message += "], ";
  }
  message += " ]";
  server.send(200, "application/json", message);
}
```

Listing 3. JSON generation function

Source code of the JavaScript function generating the summary of the LED matrix state in JSON format is shown in Listing 3. These data are generated and sent by the network only at the client's request. It also helps to reduce the traffic. The initialization code executing on the client's side is shown in Listing 4.

```
var xmlhttp;

function init() {
  xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      d = JSON.parse( this.responseText);
      for (var r = 0; r < 8; ++r) {
        for (var c = 0; c < 24; ++c) {
          document.getElementById( "led" + "_" + c + "_"
            + r).style.backgroundColor
            = (d[r][c] == 1)? "blue" : "gray";
          }
        }
      }
    };
  xmlhttp.open( "GET", "/jsonstate", true);
  xmlhttp.send();
  buildLedTable();
  }
```

Listing 4. Initialization and installation of Ajax event processor

The initialization is performed only once at the beginning of the client's work and consists of five stages. The first step is to create the object copy of XMLHttpRequest type with xmlhttp identifier. The second one is anonymous function for JSON data discrimination. At the third stage, this function is registered as onreadystatechange event processor for the xmlhttp object. At the fourth stage, the asynchronous request for JSON data from the server is performed. At the fifth one, an HTML table is built that presents the state of the LED matrix.

Source code of the function for the construction of LED matrix state representation table is shown in Listing 5. In order to increase the browser speed, the table is constructed using the functions of operation with DOM structure of HTML document [21, 22].

```
     function buildLedTable() {
       var table = document.createElement("table");
       for (var r = 0; r < 8; ++r) {
         var row = table.insertRow(r);
         for (var c = 0; c < 24; ++c) {
           var cell = row.insertCell( c);
           button = document.createElement( "button");
           button.id = "led" + "_" + c + "_" + r;
           button.innerHTML = c + "/" + r;
           button.onclick = function() {
             ledState( this.id, this.style.backgroundColor);
             };
           cell.appendChild( button);
           }
         }

document.getElementById( "ledTable").appendChild( table);
       }
```

Listing 5. Ajax optimization of LED module status table generation function

LED status display and control is performed using ledState function, the code of which is shown in Listing 6.

```
     function ledState( buttonId, buttonColor) {
       var ledData = buttonId.split( '_');
       var state = "off";
       if (buttonColor == "gray") {
         state = "on";
         }
       if (buttonColor == "blue") {
         state = "off";
         }
       xmlhttp.open( "GET", "/" + state + "?x="+ ledData[1] +
         "&y=" + ledData[2], true);
       xmlhttp.send();
       }
```

Listing 6. LED status control function

Estimation of WiFi traffic volumes after optimization. Figure 3 shows the code volume ratio for all three software architectures.

– HTML Only is the transfer of HTML code of the whole page to the client according to REST concept.

– JavaScript Optimization is the transfer of HTML code of the whole page with JavaScript optimization to the customer.

– Ajax & JSON Optimization is the transfer of HTML code of the page to the customer with Ajax & JSON optimization.

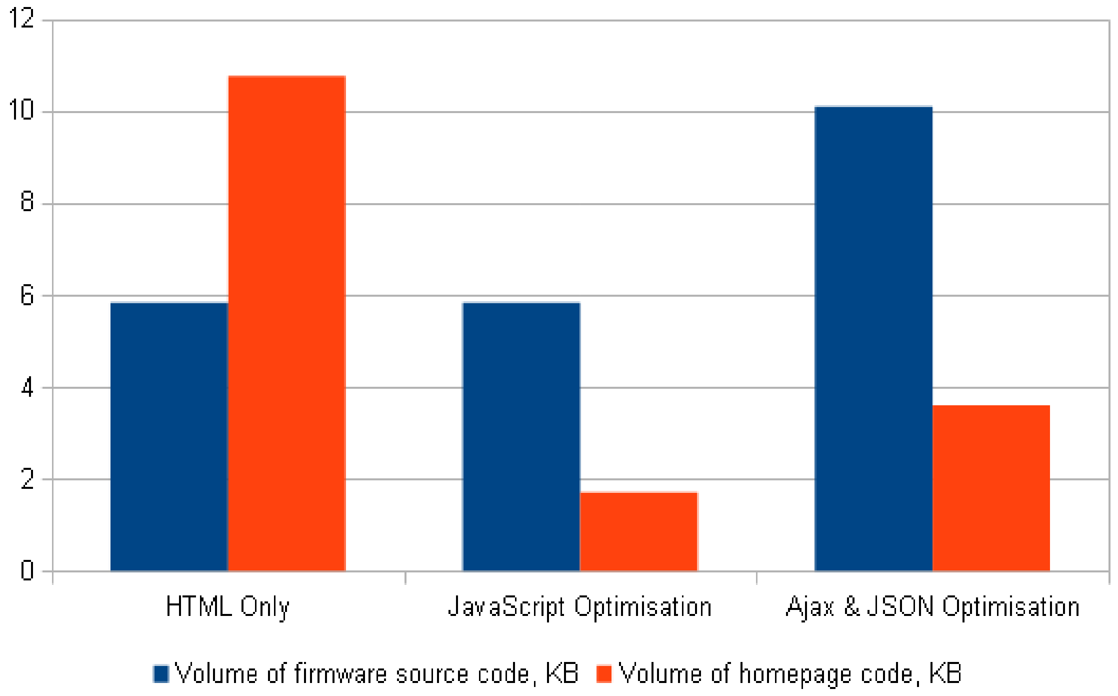
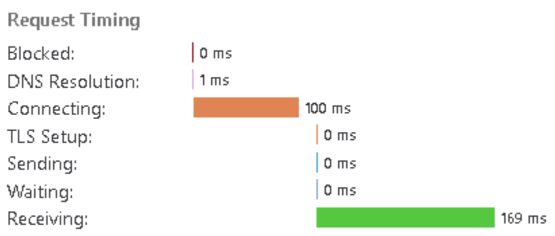


Figure 3. Program code volumes comparison

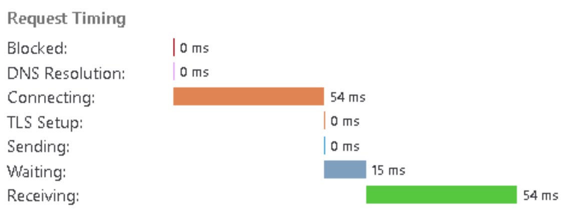
It is evident that in the second case, the sizes of both the firmware code and the HTML code are the smallest. Software architecture with Ajax & JSON optimization makes it possible to divide the client initialization process into two stages. The first stage is to transfer the home page to the client, its grammar analysis, installation of Ajax event handler, and construction of the table for displaying and controlling the status of LED matrix. The second stage is the exchange of JSON data, their grammar analysis and representation of LEDs states in the table. The time diagram of the first stage is shown in Figure 4. The time diagram of the second stage is shown in Figure 5.



Request Timing

Blocked:	0 ms
DNS Resolution:	1 ms
Connecting:	100 ms
TLS Setup:	0 ms
Sending:	0 ms
Waiting:	0 ms
Receiving:	169 ms

Figure 4. Start page loading timeline



Request Timing

Blocked:	0 ms
DNS Resolution:	0 ms
Connecting:	54 ms
TLS Setup:	0 ms
Sending:	0 ms
Waiting:	15 ms
Receiving:	54 ms

Figure 5. JSON data loading timeline

The great advantage of software architecture with Ajax & JSON optimization is that the actions of the first stage are performed only once at the beginning of the communication session. Then only the actions of the second stage are performed. Thus, with an increase in the number of requests, the proportion of static information that is transmitted by the network decreases significantly. The dynamics of reducing the share of static content is shown in Figure 6.

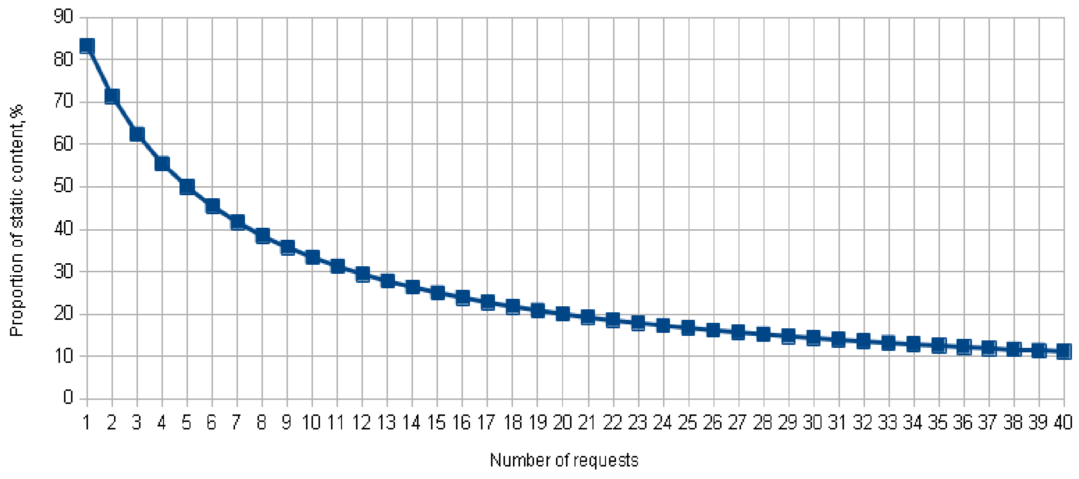


Figure 6. The dynamics of reducing the share of static content with increasing number of requests

This significantly reduces the total amount of network traffic compared to the first two software architectures. The benefits of software architecture with Ajax & JSON after the fifth request is shown in Figure 7.

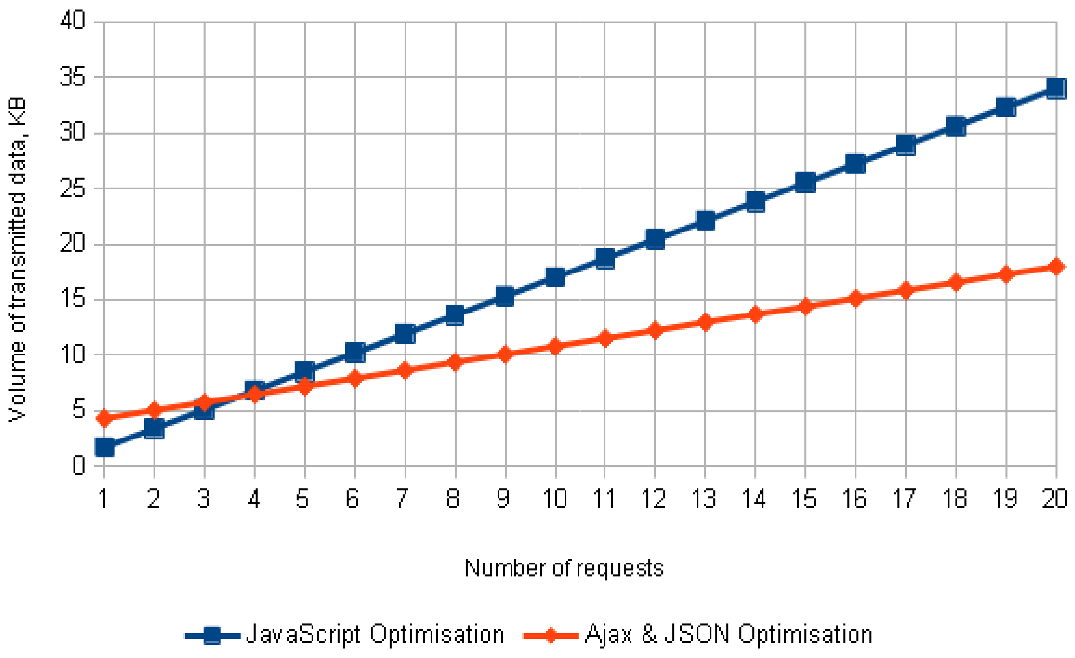


Figure 7. Comparison of generated data volumes in JavaScript and Ajax & JSON optimization

Conclusions. To study the possibilities of optimizing traffic in the IoT network at the level of software architecture and data representation, a hardware-software module for controlling the LED matrix using WiFi technology was developed. Using this module, a comparison of the network efficiency of the following software architectures was performed:

– HTML Only – transfer to the client the HTML code of the page in full according to the REST concept.

– JavaScript Optimization – transfer to the client the HTML code of the page with JavaScript optimization.

– Ajax & JSON Optimization – transfer to the client the HTML code of the page with Ajax & JSON optimization.

According to the comparison results it is shown that the software architecture based on Ajax & JSON has the highest network efficiency, significantly reducing the network traffic compared to others.

References

1. J. Fox, A. Donnellan and L. Doumen, "The deployment of an IoT network infrastructure, as a localised regional service" 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). 2019. P. 319–324. DOI: https://doi.org/10.1109/WF-IoT.2019.8767188
2. D. Moongilan, "5G Internet of Things (IOT) Near and Far-Fields and Regulatory Compliance Intricacies" 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). 2019. P. 894–898. DOI: https://doi.org/10.1109/WF-IoT.2019.8767334
3. Jim Green, IoT Reference Model. 2014. URL: http://cdn.iotwf.com/resources/72/IoT_Reference_Model_04_June_2014.pdf.
4. Overview of the Internet of things. Recommendation ITU-T Y.2060. 2014. URL: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2060-201206-I!!PDF-E&type=items.
5. FireBeetle Covers-24x8 LED Matrix (Blue). 2020. URL: https://www.dfrobot.com/product-1595.html.
6. FireBeetle Covers-24x8 LED Matrix (Blue) Product WIKI. 2020. URL: https://wiki.dfrobot.com/FireBeetle_Covers-24\%C3\%978_LED_Matrix.
7. HOLTEK HT1632C 32x8 & 24x16 LED Driver. 2020. URL: https://cdn-shop.adafruit.com/datasheets/ht1632cv120.pdf.
8. FireBeetle ESP8266 IOT Microcontroller (Supports Wi-Fi). 2020. URL: https://www.dfrobot.com/product-1634.html.
9. TL-WR702N 150Mbps Wireless N Nano Router. 2020. URL: https://www.tp-link.com/uk/home-networking/wifi-router/tl-wr702n/.
10. Y. Zhao and X. Wan, "The Design of Embedded Web System based on REST Architecture" 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). 2019. P. 99–103. DOI: https://doi.org/10.1109/IAEAC47372.2019.8997929
11. S. Malik and D. Kim, "A comparison of RESTful vs. SOAP web services in actuator networks" 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN). 2017. P. 753–755. DOI: https://doi.org/10.1109/ICUFN.2017.7993893
12. Simple Object Access Protocol (SOAP) 1.1. URL: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/.
13. Web Services Description Language (WSDL). 1.1 URL: https://www.w3.org/TR/2001/NOTE-wsdl-20010315.
14. R. A. Bahlool and A. M. Zeki, "Comparative Study between Web Services Technologies: REST and WSDL" 2019 International Conference on Innovation and Intelligence for Informatics. Computing, and Technologies (3ICT). 2019. P. 1–4. DOI: https://doi.org/10.1109/3ICT.2019.8910298
15. J. S. Zepeda and S. V. Chapa, "From Desktop Applications Towards Ajax Web Applications" 2007 4th International Conference on Electrical and Electronics Engineering. 2007. P. 193–196. DOI: https://doi.org/10.1109/ICEEE.2007.4345005
16. N. R. Dissanayake and G. K. A. Dias, "Best practices for rapid application development of AJAX based Rich Internet Applications" 2014 14th International Conference on Advances in ICT for Emerging Regions (ICTer). 2014. P. 63–66. DOI: https://doi.org/10.1109/ICTER.2014.7083880
17. X. Wang, "AJAX technology applications in the network test system" 2011 International Conference on Electrical and Control Engineering. 2011. P. 1954–1956. DOI: DOI: https://doi.org/10.1109/ICECENG.2011.6057742
18. JSON-LD 1.1. URL: https://www.w3.org/TR/json-ld11/.

19. B. Lin, Y. Chen, X. Chen and Y. Yu, “Comparison between JSON and XML in Applications Based on AJAX” 2012 International Conference on Computer Science and Service System. 2012. P. 1174–1177. DOI: https://doi.org/10.1109/CSSS.2012.297
20. G. Wang, “Improving Data Transmission in Web Applications via the Translation between XML and JSON” 2011 Third International Conference on Communications and Mobile Computing. 2011. P. 182–185. DOI: https://doi.org/10.1109/CMC.2011.25
21. Level 1 Document Object Model Specification. W3C Working Draft 20 July. 1998. Version 1.0. URL: https://www.w3.org/TR/WD-DOM/cover.html.
22. C. Y. Kang, “DOM-Based Web Pages to Determine the Structure of the Similarity Algorithm” 2009 Third International Symposium on Intelligent Information Technology Application. 2009. P. 245–248. DOI: https://doi.org/10.1109/IITA.2009.218

Список використаної літератури

1. J. Fox, A. Donnellan and L. Doumen, «The deployment of an IoT network infrastructure, as a localised regional service» 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). 2019. P. 319–324. DOI: https://doi.org/10.1109/WF-IoT.2019.8767188
2. D. Moongilan, «5G Internet of Things (IOT) Near and Far-Fields and Regulatory Compliance Intricacies» 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). 2019. P. 894–898. DOI: https://doi.org/10.1109/WF-IoT.2019.8767334
3. Jim Green, IoT Reference Model. 2014. URL: http://cdn.iotwf.com/resources/72/IoT_Reference_Model_04_June_2014.pdf.
4. Overview of the Internet of things. Recommendation ITU-T Y.2060. 2014. URL: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2060-201206-I!!PDF-E&type=items.
5. FireBeetle Covers-24x8 LED Matrix (Blue). 2020. URL: https://www.dfrobot.com/product-1595.html.
6. FireBeetle Covers-24x8 LED Matrix (Blue) Product WIKI. 2020. URL: https://wiki.dfrobot.com/FireBeetle_Covers-24\%C3\%978_LED_Matrix.
7. HOLTEK HT1632C 32x8 & 24x16 LED Driver. 2020. URL: https://cdn-shop.adafruit.com/datasheets/ht1632cv120.pdf.
8. FireBeetle ESP8266 IOT Microcontroller (Supports Wi-Fi). 2020. URL: https://www.dfrobot.com/product-1634.html.
9. TL-WR702N 150Mbps Wireless N Nano Router. 2020. URL: https://www.tp-link.com/uk/home-networking/wifi-router/tl-wr702n/.
10. Y. Zhao and X. Wan, «The Design of Embedded Web System based on REST Architecture» 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). 2019. P. 99–103. DOI: https://doi.org/10.1109/IAEAC47372.2019.8997929
11. S. Malik and D. Kim, «A comparison of RESTful vs. SOAP web services in actuator networks» 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN). 2017. P. 753–755. DOI: https://doi.org/10.1109/ICUFN.2017.7993893
12. Simple Object Access Protocol (SOAP) 1.1. URL: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/.
13. Web Services Description Language (WSDL). 1.1 URL: https://www.w3.org/TR/2001/NOTE-wsdl-20010315.
14. R. A. Bahlool and A. M. Zeki, «Comparative Study between Web Services Technologies: REST and WSDL» 2019 International Conference on Innovation and Intelligence for Informatics. Computing, and Technologies (3ICT). 2019. P. 1–4. DOI: https://doi.org/10.1109/3ICT.2019.8910298
15. J. S. Zepeda and S. V. Chapa, “From Desktop Applications Towards Ajax Web Applications” 2007 4th International Conference on Electrical and Electronics Engineering. 2007. P. 193–196. DOI: https://doi.org/10.1109/ICEEE.2007.4345005
16. N. R. Dissanayake and G. K. A. Dias, “Best practices for rapid application development of AJAX based Rich Internet Applications” 2014 14th International Conference on Advances in ICT for Emerging Regions (ICTer). 2014. P. 63–66. DOI: https://doi.org/10.1109/ICTER.2014.7083880
17. X. Wang, «AJAX technology applications in the network test system» 2011 International Conference on Electrical and Control Engineering. 2011. P. 1954–1956. DOI: https://doi.org/10.1109/ICECENG.2011.6057742
18. JSON-LD 1.1. URL: https://www.w3.org/TR/json-ld11/.
19. B. Lin, Y. Chen, X. Chen and Y. Yu, “Comparison between JSON and XML in Applications Based on AJAX” 2012 International Conference on Computer Science and Service System. 2012. P. 1174–1177. DOI: https://doi.org/10.1109/CSSS.2012.297
20. G. Wang, “Improving Data Transmission in Web Applications via the Translation between XML and JSON” 2011 Third International Conference on Communications and Mobile Computing. 2011. P. 182–185. DOI: https://doi.org/10.1109/CMC.2011.25

21. Level 1 Document Object Model Specification. W3C Working Draft 20 July. 1998. Version 1.0. URL: https://www.w3.org/TR/WD-DOM/cover.html.
22. C. Y. Kang, "DOM-Based Web Pages to Determine the Structure of the Similarity Algorithm" 2009 Third International Symposium on Intelligent Information Technology Application. 2009. P. 245–248. DOI: https://doi.org/10.1109/IITA.2009.218

УДК 004.772

ОПТИМІЗАЦІЯ ТРАФІКУ У WIFI МЕРЕЖАХ ДЛЯ ІНТЕРНЕТУ РЕЧЕЙ

В'ячеслав Старченко

Чорноморський національний університет імені Петра Могили, Миколаїв, Україна

Резюме. *Однією з головних проблем сучасних мереж IoT є великий обсяг автоматизованого трафіку, що генерується їх вузлами. Це створює значне навантаження на сучасні мережі зв'язку, яке з плином часу буде тільки збільшуватися. Одним із шляхів подолання цієї проблеми є оптимізація структури даних та покращення методів їх збирання, передавання та опрацювання. Метою даного дослідження є оптимізація трафіку в мережі IoT на рівні програмної архітектури та представлення даних. Об'єктом дослідження обрано світлодіодну матрицю FireBeetle Covers-24 x 8 LED Matrix ESP32, керування якою здійснюється за допомогою контролера HOLTEK HT1632C, яке відбувається через WiFi-інтерфейс з IoT-мережею, реалізований на базі мікроконтролера ESP8266. Предметом дослідження є процес оптимізації програмного коду шляхом вибору найефективнішої програмної архітектури. Розглянуто три найпоширеніших програмних архітектури та наведено приклади їх реалізації. Першою розглянуто архітектуру на базі REST-технології. Ця технологія є найпопулярнішою та розповсюдженою завдяки простоті концепції та реалізації. Але суттєвий її недолік полягає у значному перевантаженні лінії зв'язку великим обсягом однотипної службової інформації. Зменшення обсягу службової інформації за рахунок оптимізації коду HTML-сторінки за допомогою мови JavaScript продемонстровано на прикладі другої програмної архітектури. Але така оптимізація не дозволяє повністю розділити статичну та динамічну складові інформації, що передається лінією зв'язку. Такий розподіл легко можна здійснити у рамках програмної архітектури на базі Ajax & JSON, приклад якої наведено третім. Великою перевагою такої архітектури є те, що статична складова передається мережею лише один раз на початку сеансу зв'язку. Потім передається лише динамічна складова. За результатами випробування розробленого апаратно-програмного модуля та порівняння обсягів згенерованих даних, що передаються WiFi-мережею, показано, що програмна архітектура на базі Ajax & JSON має найбільшу мережеву ефективність, значно зменшуючи мережевий трафік порівняно з іншими.*

Ключові слова: *бездротова технологія, програмна архітектура, Internet of Things, WiFi, JavaScript, Ajax, REST, JSON.*

https://doi.org/10.33108/visnyk_tntu2021.04.131 *Отримано20.11.2021*