Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 7, No. 2, June 2019, pp. 279~288
ISSN: 2089-3272, DOI: 10.11591/ijeei.v7i2.979

A bridge between legacy wireless communication systems and internet of things

Samer Jaloudi
Department of Information and Communication Technology, Al Quds Open University, Palestine

Article Info

Article historys:

Received Jan 24, 2019
Revised Apr 30, 2019
Accepted May 31, 2019

Keywords:

HTTP
Internet of things
Interoperability
JSON
MQTT
Protocols
Scalability
SDR
Software-defined radio
XMPP

ABSTRACT

The software-defined radio (SDR) is a flexible platform that can adapt to various wireless telecommunication frequencies. It is able to provide a reconfigurable communication infrastructure for wireless systems. Hence, SDR is proposed here as a bridge between legacy wireless communication systems and the Internet of Things (IoT) via standard telecommunication protocols. The standard protocols are hypertext transfer protocol (HTTP), simple mail transfer protocol (SMTP), and message queuing telemetry transport (MQTT). Data collected from legacy wireless systems have been formatted via JavaScript object notation (JSON) for interoperability and categorized according to the application and the communication pattern. The extracted data are then transferred over MQTT for machine-to-machine (M2M) communication, over SMTP for machine-to-human (M2H) notification, and over HTTP for human-to-machine (H2M) communication. However, received audio signals from FM-based broadcasting stations have been transferred to the Internet servers over extensible messaging and presence protocol (XMPP), in live audio streaming. The objective is to introduce an SDR-IoT bridge that is inexpensive, scalable, and interoperable. The analyses show that the environment has good-performance, and can be used for many applications of smart city sectors, for Internet Radio, and for Internet-based monitoring of airplanes and vessel navigation.

Corresponding Author:

Samer Jaloudi,
Department of ICT,
Al Quds Open University,
Rafedia, P. O. Box 468, Nablus, West Bank, Palestine.
Email: sgalodi@qou.edu

1. INTRODUCTION

The software-defined radio (SDR) [1-3] is a reconfigurable wireless platform that consists of two parts, the hardware part, and the software part. Most of the operations of SDR are executed in the software part, and future upgrades can be applied without hardware changes. Indeed, the SDR has been employed in several applications and, in many cases, introduced inexpensive, easy to use and program platforms. Furthermore, it introduces a modular and scalable solution, in addition to the reduced physical dimensions of the developed system. The SDR architecture provides a reconfigurable platform that minimizes costs, physical size and weight [4, 5], and provides flexibility and modularity required for developing prototypes for communication systems.

Legacy communication systems below 1.1 GHz include, but are not limited to, automatic dependent surveillance-broadcast (ADS-B), automatic identification system (AIS), legacy monitoring systems via 915 MHz and 433 MHz frequency bands, FM-based broadcasting systems such as wireless microphones and Walkie-talkie devices, etc. The SDR is successfully employed for receiving the signals of these legacy communication systems; ADS-B in [6, 7], AIS in [8, 9], and FM signals in [10, 11]. However, authors of these research papers did not transfer the received data over the Internet. In this paper, SDR is employed jointly with

Internet protocols for monitoring purposes, while providing interoperability via standard Internet of Things (IoT) protocols. In addition, the solution addresses some issues solved traditionally by SDR such as cost, physical dimensions, modularity, and scalability. Hence, receiving data from some legacy wireless communication systems are proposed here using SDR-based hardware and software platforms integrated, via standard Internet protocols, with online-based IoT servers. The standard protocols proposed here are message queuing and telemetry transport (MQTT) [12-14] for machine-to-machine (M2M) communication, hypertext transfer protocol (HTTP) [15] for human-to-machine (H2M) communication, and simple mail transfer protocol (SMTP) [16] for machine-to-human (M2H) notification. For interoperability, JavaScript object notation (JSON) [17] is used to format the extracted data from the legacy communication systems, and then transfer them to Internet-based servers. However, the extensible message and presence protocol (XMPP) [18] is employed to transfer the extracted audio signals from the FM-based broadcasting stations over the Internet, in live audio streaming. Using this SDR-IoT platform, an inexpensive and scalable solution is achieved, thanks to the inexpensive RTL-SDR [1, 10, 11], costing less than $20, and the accompanying free software tools. Moreover, The analysis of the MQTT latency and the XMPP bandwidth show a good-performance to connectivity with the Internet while maintaining interoperability.

The remainder of the paper is organized as follows. Section II reviews standard protocols employed for IoT connectivity in this paper, and the related network model for interoperability. Section III introduces the SDR platform used in this paper, including the test environment and two practical scenarios. While section IV introduces the results and analyses, section V concludes the paper.

2. NETWORK PROTOCOLS AND MODEL

Two subsections are presented here. The first subsection introduces the standard IoT protocols and a comparison between their overheads. In the second subsection, a network model based on the IEEE model is introduced for solving the interoperability problem.

2.1. Standard internet protocols

In this subsection, standard Internet protocols used in this paper are introduced, namely, MQTT, HTTP, XMPP, JSON, and SMTP. Since MQTT is proposed for M2M communications and HTTP for H2M communications, overhead analyses of both protocols are conducted and compared with that of the XMPP, and illustrated in Figure 1.

On the wire telecommunication protocol, the open standard MQTT is dedicated to constrained devices, and transfers small binary messages using the asynchronous communication pattern. It is a publish-and-subscribe, one-to-many, and broker-based protocol. A publisher produces events in a form of messages and subscribers consume the messages. The publisher and the subscriber are clients, and the broker is the server. To produce a 5-byte message from a publisher and to consume it by one subscriber, 115 bytes are exchanged between both clients and the server. The length of the exchanged messages depends on the topic length. While, the topic length is fixed to 11 bytes, the percentage overhead of the total protocol message is calculated using (1):

$$Overhead\ (\%) = \frac{Overhead}{Overhead + Payload} \times 100\% \quad (1)$$

The same equation is used for the HTTP and XMPP, for the same purpose as shown in Figure 1, which demonstrates the percentage of the protocol overhead compared to the payload for the three protocols. The graph is generated for a payload of 0 bytes to 2048 bytes. The minimum protocol overhead is taken as a fixed value of 105 bytes, and the connection already has been established. The curve indicates that MQTT is suitable for constrained IoT devices frequently send small-sized messages.

The HTTP is a request-and-response, one-to-one (peer-to-peer), and brokerless protocol. The client issues a request via one of its commands and the server replies with document-based, readable-text response. A typical GET command requires 82 bytes of header, and a typical response requires 290 bytes of header, a total of 372 bytes, which is used in (1) to plot the HTTP curve of Figure 1.

The XMPP supports both modes; the request-and-response, and the publish-and-subscribe, in addition to transfer text, audio and video signals in chat sessions. In publish-and-subscribe mode, the overhead between the client and the server is around 1840 bytes [19], which is used in (1) to plot the XMPP curve of Figure 1. As a result, the XMPP has the highest overhead of them all. Hence, it will not be used for M2M, H2M, or M2H communications. However, it will be used to transfer audio signals extracted from legacy FM-based broadcasting stations, to an Internet-based XMPP server.

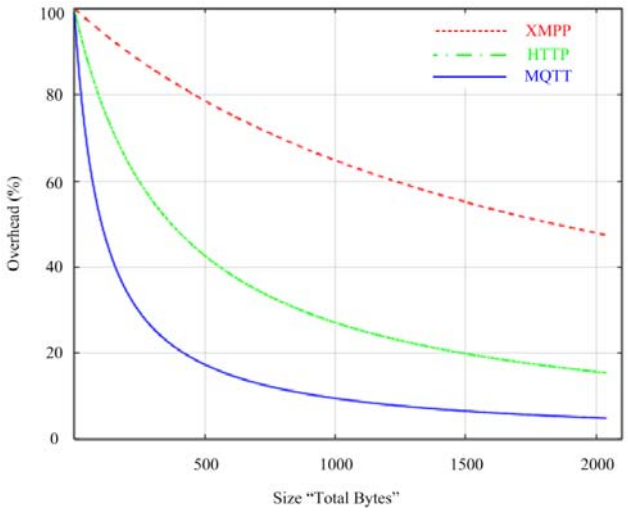

Figure 1. Overheads of protocols XMPP, HTTP, and MQTT

For lightweight exchange of data between browsers and a server, the W3C [20] has standardized JSON. The standard provides RESTful text-based communications with no complicated parsing. The underlying communication protocol is the HTTP. The JSON format is also used in this paper for interoperability; with SMTP for M2H notification and with MQTT for M2M communication. For Email-based M2H notifications, the SMTP is proposed here, where data are extracted from the legacy wireless communication systems via RTL-SDR, formatted in JSON, and then transferred to interested subscribers. Since Emails are slow, this service is proposed for noncritical warnings and alarms notifications.

2.2. Network model for interoperability

A network model, shown in Figure 2, is proposed here to solve the problem of interoperability based on the IEEE model. The SDR works in the physical layer, where wireless signals are received, demodulated, and useful data extracted. Useful data include mainly measurements frames and audio signals. To transfer these data over the Internet, an intermediate gateway (IoT clients) is proposed here, which communicates with the SDR software via User Datagram Protocol (UDP), and transfers these data over the IoT standard. The standard Internet protocols are XMPP for the extracted audio signals, JSON over MQTT for M2M communication, JSON over HTTP for H2M communication, and JSON over SMTP for M2H notifications. This arrangement solves interoperability issues in the system where various nodes transfer different formats of data.

For web browser–based monitoring, JSON introduces its ASCII-based text formatted in JavaScript objects that are transferred over HTTP via TCP port 80 or SMTP via TCP port 587. This arrangement introduces good level of interoperability for people that are watching behind their mobile phones and tablets. However, for transferring data frames of AIS and ADS-B applications over the Internet, and due to the varying bandwidth and the unreliable nature of such networks, the communication protocol must be lightweight, easy to program, flexible, and reliable. The publish-and-subscribe nature of MQTT makes this protocol suitable for this mission. Thus, the MQTT is proposed here in two levels of communication-infrastructure, namely the publisher (gateway) and the consumer (website).

The MQTT uses UTF-8 encoding format, in binary representation. However, the protocol does not specify the details of the object representation, and hence, interoperability is not fulfilled. JSON introduces its text-based representation of objects for enhancing interoperability, and therefore, it is adopted here to transfer extracted data in the payload of the MQTT. Using this arrangement, signals received from sensors, ADS-B, AIS, etc. are demodulated, analyzed, useful information extracted, data frames are formatted in JSON, transferred in the payload of MQTT using TCP port 1883, then over IP in the network layer, and over Wi-Fi or Ethernet in the physical layer. Concerning audio signals, XMPP is the suitable protocol, which uses TCP port 5222. More details will be given in Section 3 about the protocol used by the XMPP for transferring audio signals.

Measurements, ADS-B, AIS, etc. Data formatted in JSON			FM station Audio Signals
M2H communication	H2M communication	M2M communication	Archiving Internet-Radio
SMTP	HTTP	MQTT	XMPP
TCP Port 587	TCP Port 80	TCP Port 1883	UDP Port 16384-32767
IPv4, or IPv6			
Ethernet: IEEE 802.3 Wi-Fi: IEEE 802.11			

Figure 2. Network model based on the IEEE model for interoperability

3. THE SDR-IoT ENVIRONMENT

There exist many SDR devices in the market, ranging from expensive units, like those of USRP [21, 22] to inexpensive units, such as the USB-based RTL-SDR dongles. These dongles are in use as PC-based receivers of digital video broadcasting terrestrial (DVB-T) signals, and their scan of RF signals is limited in the frequency range of 24 MHz to 1766 MHz [23]. Moreover, they have limited bandwidth of 2 MHz, limited frequency range, and can't transmit signals. In fact, the field of application depends on the platform's capabilities. For example, the very low-cost RTL-SDRs are suitable for hobby and test below 1766 MHz [1, 10, 11], and the USRP platforms for cellular networks [24], and agile communication systems [25]. The software part is either open source or commercial. The MATLAB/SIMULINK is an example of well-supported proprietary software, and the free open source software (FOSS) called GNU Radio Companion (GRC) [26] is an example of gratis software. For this study, free software tools, including the GRC, and the inexpensive RTL-SDR are used for receiving the signals of legacy wireless communication systems that use frequencies below 1.1 GHz.

For this purpose, the system architecture is developed and illustrated via two scenarios as shown in Figure 3 and Figure 4. The first scenario is for receiving the signals of FM-based broadcasting stations via the RTL-SDR. This scenario is developed here to transfer audio signals using the XMPP IoT protocol to an Internet-based XMPP server, as shown in Figure 3. Therefore, for Internet radio, for archiving purposes, and for transferring the audio signals of a specific broadcasting FM station over the Internet using IoT standards, SDR software based on the GRC, and an SDR hardware based on the RTL-SDR have been employed as illustrated in Figure 5. The output of the SDR software is then transferred to a gateway using a UDP client via port 7777.

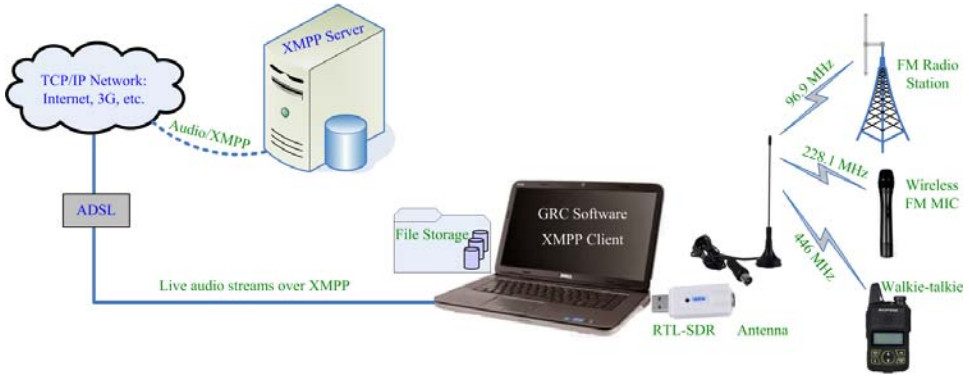

Figure 3. First scenario: receiving audio signals of FM-based broadcasting stations via an SDR, then signals are transferred over the Internet to an XMPP server

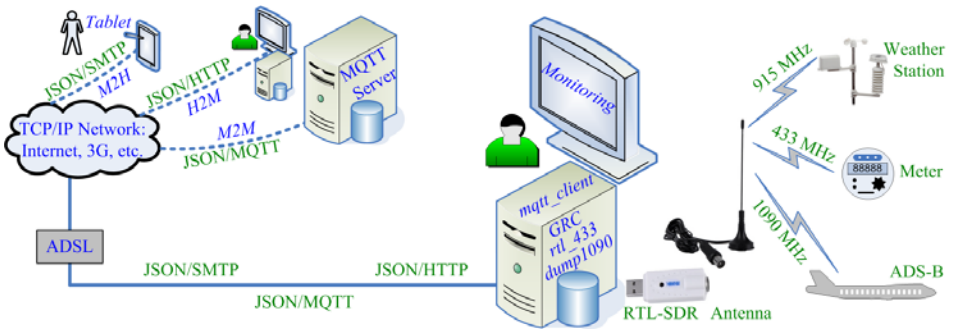

Figure 4. Second scenario: receiving measurements, monitoring data of legacy sensor nodes, and ADS-B signals using an SDR, then formatted in JSON and transferred over the Internet to protocols' servers

The design of Figure 5 starts with a signal source in the RF range, which is the block named RTL-SDR Source with a sampling frequency of 2 MHz. The signal flows from the source to a block of IF LPF (Intermediate frequency Low Pass Filter), with a decimation factor of 10, to Rational Resampler, and finally, to the Wideband FM (WBFM) receiver block. To transfer the audio signals to the IoT protocol client, a UDP block is used with port 7777.

Since the bandwidth of the audio channel is 200 kHz, the decimation value of the low pass filter (LPF) is 10, which reduces the sample rate from 2 MHz to 200 kHz, to become compatible with the audio signal. At the output, the sample rate of the audio signal is 48 kHz, thus the audio decimation value is 10, which reduces the sample rate of the WBFM block from 480 kHz to 48 kHz. A connecting block between the LPF and WBFM blocks is needed to recalibrate the sample rate between both blocks, which is the rational resampler block. It increases the sampling rate from 200 kHz by a factor of 12, and decreases it by a value of 5 so that it becomes compatible with the sample rate value of the WBFM receiver, which is 480 kHz.

The same scenario is applicable to FM-based wireless microphones in VoIP sessions, and Walkie-talkie devices used in police, firefighting, and emergency departments. The difference is in the baseband frequency, which is 446 MHz for the Walkie-talkie, in the UHF band, 228.1 MHz for the wireless microphone, in the VHF band, and 96.9 MHz for the broadcasting Radio station in the VHF band as well. Thus, the difference is in the software, namely in the slider frequency range (WX GUI Slider block) that determines the exact receiving center-frequency.

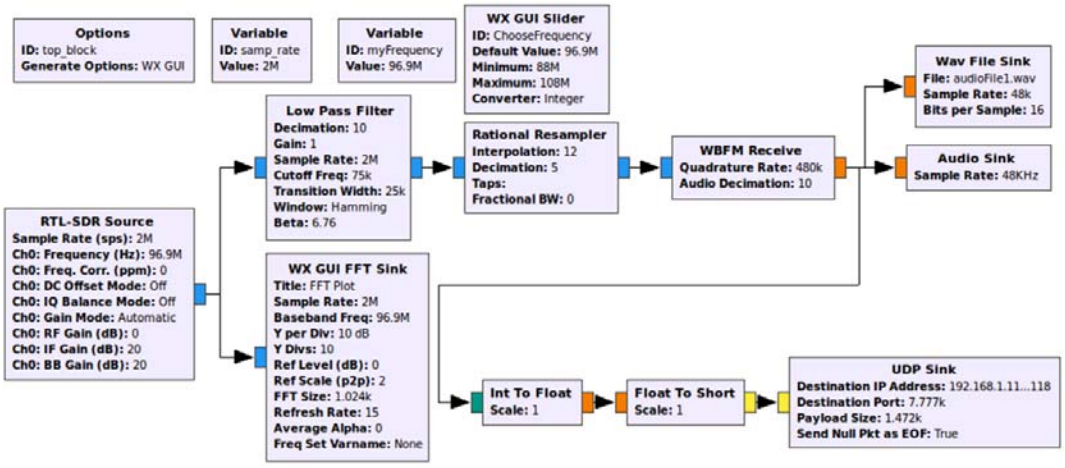

Figure 5. Block diagram of the GRC-based software for receiving signals broadcasted by an FM Radio station using RTL-SDR. The extracted audio signals are then transferred to an Internet-based XMPP server. ID: identification; Freq.: frequency; ppm: pulse per minute; EOF: end of file; Fractional BW: fractional bandwidth; Freq. Corr. (ppm): frequency correction factor in ppm; IQ: raw I-Q data; RF: radiofrequency; IF: intermediate frequency; BB: overall baseband gain of the device; sps: samples per second; Taps: automatic value; FFT: fast Fourier transform

In the second scenario, shown in Figure 4, signals of monitoring devices based on legacy wireless communications were received using the RTL-SDR device. Sensors that use legacy wireless communication systems based on the 433 and 915 MHz frequencies for measurements and monitoring benefit from the RTL-SDR platform to format extracted data in JSON then transfer them using one of the Internet protocols, HTTP, SMTP, or MQTT. Therefore, these sensors become things in the IoT cloud. Using the same scenario, signals carrying air traffic information about airplanes passing over the region, have been received, extracted from the ADS-B frames, formatted in JSON, and then transferred in the payload of an MQTT message.

For the 433 MHz-based devices, a free software tool, called rtl_433 [27], is used to detect 433 MHz signals transmitted by a legacy sensor node, shown in Figure 5. The tool is able to extract the hexadecimal frame of the transmitted signal. The output of this tool is then collected via the gateway (IoT clients that written in Java), which analyzes the frame format, stores locally the measurement values, and then transfers these values in the payload of the JSON format over HTTP, when polled by users with portable devices. Since HTTP is polling-based protocol (request-response), this solution is unable to provide notifications, and hence, measurements are encapsulated in JSON format and transferred over SMTP. This solution provides M2H Email notifications via SMTP, or H2M monitoring via HTTP. Another scalable, modular, and interoperable solution, achieved thanks to the SDR platform integrated with a standard IoT protocol, which is JSON/HTTP, or with a standard Internet protocol, which is JSON/SMTP.

The free software tool *dump1090* [28] is employed for receiving ADS-B signals. The tool is able to extract the hexadecimal frames of the transmitted signals by airplanes, and to analyze them to obtain useful information. The output of this tool is then collected via the Java-based IoT protocol client, which stores locally the useful information, formats the required data in JSON, and transfers these formats in the payload of an MQTT message over the Internet to an MQTT server available from the website of *hivemq.com* for testing.

4. RESULTS AND ANALYSIS

The performance of the system is mainly dependent on the performance of the used protocol, since the IoT client receives the data from the RTL-SDR, structures them in the protocol's format of XMPP or JSON over SMTP, MQTT, or SMTP, and then transfers the message to an Internet-based server. The SMTP is proposed in this study for noncritical monitoring of alarms and warnings (M2H notifications). It uses public and Internet-based relays (message transfer agents) that increase the delay between endpoints. An e-mail delivery typically takes several seconds, and in most cases, the process takes less than a minute. More details on Email and SMTP analyses can be found in [29-31]. The HTTP is proposed here for H2M communication, and hence human-generated requests for retrieving information will not be analyzed because JSON/HTTP communications have been thoroughly analyzed, for example, in [32-36]. However, MQTT is proposed for M2M communication, and therefore, latency tests are conducted in the first subsection (4.1). Since the XMPP is proposed for live audio streaming, bandwidth requirements are calculated in the second subsection (4.2).

4.1. MQTT latency

The MQTT is proposed in this study as an IoT protocol for M2M communication, and thus, three latency tests are conducted in this subsection. The first latency test is conducted via three online servers. Three tests are conducted to assess the roundtrip delay of online servers (brokers) including the *HiveMQ* test server available online at *broker.hivemq.com*, the *Mosquitto*-based test server available online at *test.mosquitto.org*, and the test server of *Moquette* available online at *broker.moquette.io*. The results are shown in Figure 6, Figure 7, and Figure 8. The clients that connect to these online servers start counting the round-trip time (RTT) from creating the socket to closing it, which includes the time needed for receiving the frame from the RTL-SDR, formatting the message in JSON, connecting with the server without logon process, and disconnecting from the server, in addition to the network latency. The *HiveMQ* server has the less delay, and the *Moquette* server has the highest delay. In fact, MQTT is a broker-based protocol, which may face performance issues, especially when the number of simultaneous connections increases, as shown in Figure 6. In this case, MQTT servers managed in clusters help solving performance issues.

Since the *HiveMQ* server has the less RTT, it is used for the second test, where one-hundred, fixed-sized messages of 5-bytes were published (sent) over the Internet to the server. One thread publishes one-hundred messages successively, two concurrent threads each publishes fifty messages, four concurrent threads each publishes twenty-five messages, five concurrent threads each publishes twenty messages, and ten concurrent threads each publishes ten messages. The result is illustrated in Figure 7, which shows that ten concurrent threads, each sends ten messages, consume the less time per message. This is because one publisher sends one-hundred messages successively, uses one TCP route, and hence any message faces a delay on the route influences the other messages in the queue. With ten concurrent threads, the ten TCP routes are different, each route handles ten successive messages, and thus the RTT per message is 21.49 ms. However, the RTT of

each message is 205.44 ms with one publisher that publishes one-hundred messages successively. This test indicates that the data received by the RTL-SDR from legacy communication systems should be formatted in JSON, distributed on threads, and then published by the MQTT publisher concurrently, instead of successively. Therefore, the received data from weather stations, meters, and ADS-B systems have been formatted in JSON, subdivided into smaller messages, and published concurrently via Java threads.

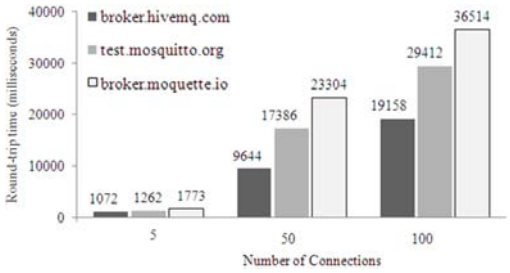

Figure 6. MQTT latency: comparing the latency of three online servers

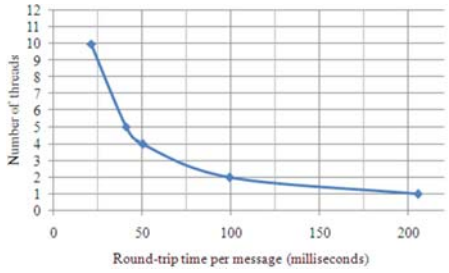

Figure 7. MQTT latency: number of threads versus RTT per message

The third test is also conducted with the Internet-based *HiveMQ* server as well, where five cases were tested. One thread publishes one message of payload 10 bytes, twenty bytes, forty bytes, fifty bytes, and one-hundred bytes. The result is illustrated below in Figure 8, which indicates that the latency is always between 200 ms and 300 ms, which is an indication of the good performance of the developed environment.

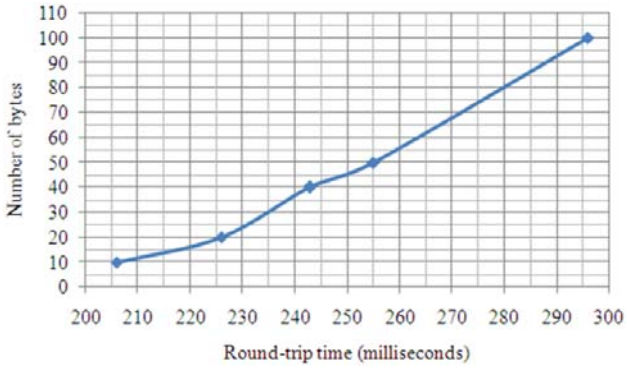

Figure 8. MQTT latency: number of bytes versus RTT

As shown in Figure 1, producing one message by one publisher and consuming it by one subscriber, requires around 100 bytes of overhead. Figure 8 shows that a message of 100-bytes' payload requires 300 ms of RTT. Hence, a message of payload 100 bytes requires around 300 ms of RTT and 50% of overhead.

4.2. XMPP bandwidth

There are mainly two XMPP extensions for audio sessions, Jingle [37] and Jingle RTP Session [38]. Jingle is a collection of protocols for initiating, maintaining, and terminating peer-to-peer (P2P) multimedia sessions over the Internet. When a session is initiated, media such as voice chat is exchanged using the Real-time Transport Protocol (RTP) [39]. Usually, the process for negotiating a Jingle RTP session follows 6 steps [37], arranged in 6 XML messages. The initiator starts the process by sending a *session-initiation* message to the responder. The responder acknowledges the message reception by sending an *ack* message, followed by a *session-accept* message. The initiator acknowledges the *session-accept* by sending *ack* message. Now, both parties can exchange media using RTP. Any party is able to terminate the session by sending a *session-terminate* message. The other party acknowledges the reception of termination requirement by sending an *ack* message that terminates the session.

The RTP uses one of three time-intervals, which are ten milliseconds, twenty milliseconds (the default), and thirty milliseconds. The payload is compatible with each interval, which is 80 bytes, 160 bytes, or 240 bytes respectively that guarantees the 64 kbps bandwidth in all cases. In the first case, the time interval is 10 ms and hence the number of packets per seconds (pps) is 100. In the second case, the time interval is 20 ms and hence the pps is 50. Finally, in the third case, the time interval is 30 ms and hence the pps is 33.33. The required bandwidth in each case is calculated using (2). Since RTP is using UDP/IP socket communication, the total header is calculated based on the header of the three protocols. The RTP uses 12 bytes of header, the UDP uses 8 bytes of header, and the IP uses 20 bytes of header. The Ethernet header is not part of the equation, because the ADSL is used in this case to pass the RTP packets over the Asynchronous Transfer Mode (ATM) of the WAN to the Internet-based server.

$$Bandwidth = \left[Header + Payload\right] \times 8 \times Packets \quad (2)$$

The result is plotted in Figure 9, where (a) illustrates the size (number of bytes) versus the time interval while demonstrating the header compared to the payload, and (b) shows the bandwidth according to (2) for each time interval. The maximum bandwidth required by the XMPP for transferring audio signals to an Internet server is 96 kbps, which is another indication of the good performance of the developed scenario. As mentioned above and in all cases, the bandwidth is always fixed to 64 kbps if no headers were considered.

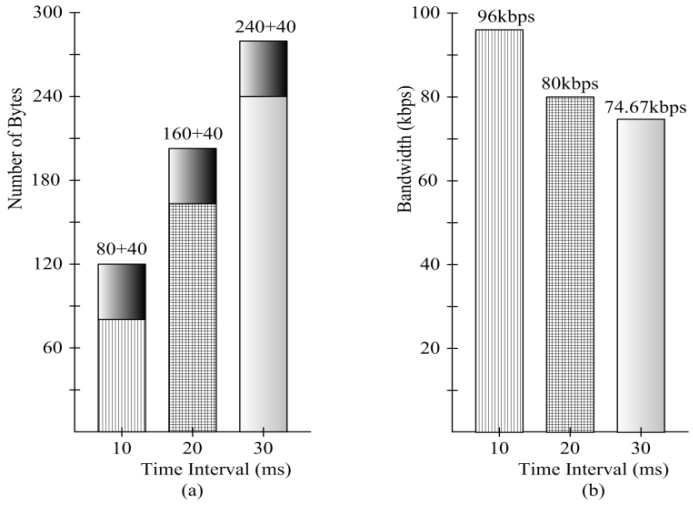

Figure 9. RTP comparisons, (a) total headers compared to the payloads versus the time interval, and (b), the bandwidths versus the Time interval

5. CONCLUSION

The IoT has been employed successfully in many studies for agriculture [40, 41], industry [42], smart cities [43], among other applications, and interoperability is solved, for example, in [44-46]. In this paper, the IoT is proposed as an interoperable platform by introducing its standard protocols to transfer data from legacy communication systems over the Internet. Signals of legacy wireless communication systems were received via an SDR platform, and hence costs and physical dimensions are reduced, scalability and modularity are achieved, and flexibility is ensured. However, the connectivity with the IoT requires a standard telecommunication protocol and format that provides interoperability. Hence, JSON is used as a standard format for interoperability, and protocols SMTP, MQTT, and HTTP are used respectively for M2H, M2M, and H2M communications. The developed SDR-IoT environment receives the signals, extracts the frame, formats the data in JSON, and then transfers them over a standard protocol. However, extracted audio signals are transferred over XMPP to an Internet-based XMPP server. The analysis of the MQTT latency and XMPP bandwidth show good performance to connectivity with Internet while providing interoperability. The developed environment can be used in many applications for transferring data from wireless systems to Internet-based servers in monitoring mode. Such applications include, but are not limited to smart city sectors, Internet Radio, Internet-based AIS and ADS-B, and voice over IP (VoIP) in audio conferencing systems.

REFERENCES

[1] R. W. Stewart, et al., "A low-cost desktop software defined radio design environment using MATLAB, simulink, and the RTL-SDR," *IEEE Communications Magazine*, vol. 53(9), pp. 64-71, Sep 2015.

[2] C. Belisle, V. Kovarik, L. Pucker, and M. Turner., "The Software Communications Architecture: Two Decades of Software Radio Technology Innovation," *IEEE Communications Magazine*, vol. 53(9), pp. 31-37, Sep 2015.

[3] T. Ulversoy., "Software Defined Radio: Challenges and Opportunities," *IEEE Communications Surveys & Tutorials*, vol. 12(4), pp. 531-550, May 2010.

[4] M. B. O'Neill, W. P. Millard, B. M. Bubnash, R. H. Mitch, and J. A. Boye., "Frontier Radio Lite: A Single-Board Software-Defined Radio for Demanding Small Satellite Missions," in *the 30th annual AIAA/USU Conference on Small Satellites*. [Available] https://digitalcommons.usu.edu/smallsat/2016/TS7Communication/2.

[5] Y. Kuo, P. Pannuto, T. Schmid, and P. Dutta., "Reconfiguring the software radio to improve power, price, and portability," in *Proc. ACM Conf. Embedded Network Sensor Systems, Toronto, Ontario*, pp. 267-280, 2012.

[6] R. N. Pahlevy, A. D. Prasetyo, and Edwar., "Nanosatellite ADS-B Receiver Prototype for Commercial Aircraft Detection," in *2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), Bandung, Indonesia*, pp. 6-12, 2018.

[7] Y. Zhang, K. Wang, and H. Long., "SDR-Based ADS-B with Dual-Frequency for UAS: A Universal Design," in *IEEE CIT, IUCC, DASC, PICOM Conference, Liverpool, UK*, pp. 59-63, 2015.

[8] J. A. Larsen and H. P. Mortensen, "In Orbit Validation of the AAUSAT3 SDR based AIS receiver," in *IEEE 6th International Conference on Recent Advances in Space Technologies (RAST), Istanbul, Turkey*, pp. 487-491, 2013.

[9] F. Mazzarella, M. Vespe, D. Tarchi, G. Aulicino, and A. Vollero., "AIS reception characterisation for AIS on/off anomaly detection," in *IEEE 19th International Conference FUSION, Heidelberg*, pp. 1867-1873, 2016.

[10] M. Saber, H. K. Aroussi, A. El Rharras, and R. Saadane., "Raspberry Pi and RTL-SDR for Spectrum Sensing based on FM Real Signals," in *IEEE 6th International Conference on Multimedia Computing and Systems (ICMCS), Rabat, Morocco*, pp. 1-6, 2018.

[11] S. Jaloudi., "Software-Defined Radio for Modular Audio Mixers: Making Use of Market-Available Audio Consoles and Software-Defined Radio to Build Multiparty Audio-Mixing Systems," *IEEE Consumer Electronics Magazine*, vol. 6(4), pp. 97-104, Oct 2017.

[12] "Message Queuing and Telemetry Transport (MQTT) Version 3.1.1," *ISO/IEC Standard PRF 20922*, 2016.

[13] "Message Queuing and Telemetry Transport (MQTT) Version 3.1.1," *OASIS Standard* ----, 2014.

[14] A. Luoto, and K. Systä, "Fighting network restrictions of request-response pattern with MQTT," *IET Software*, vol. 12(5), pp. 410-417, Oct 2018.

[15] "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," *IETF Standard RFC7231*, 2014.

[16] "Simple Mail Transfer Protocol," *IETF Standard RFC5321*, 2008.

[17] "The JavaScript Object Notation (JSON) Data Interchange Format," *IETF Standard RFC7159*, 2014.

[18] "eXtensible Message and Presence Protocol (XMPP)," *IETF Standard RFC6120*, 2011.

[19] P. Millard, P. Saint-Andre, and R. Meijer, XEP-0060: Publish-Subscribe. [Available] https://xmpp.org/extensions/xep-0060.html. 2018.

[20] JSON Introduction. W3Schools Tutorials. [Available] http://www.w3schools.com/js/js_json_intro.asp. 2017.

[21] The Universal Software Radio Peripheral (USRP™). National Instruments, [Available] https://www.ettus.com/. 2016.

[22] A. Nika, Z. Zhang, B. Y. Zhao, and H. Zheng., "Toward Practical Spectrum Permits," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3(1), pp. 112-122, Mar 2017.

[23] Realtek, RTL2832U. realtek.com.tw. [Available] http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PFid=35&Level=4&Conn=3&ProdID=257. 2017.

[24] G. Soni and G. Megh, "Experimental investigation of spectrum sensing for LTE frequency band based on USRP 2920/VST 5644," in *IEEE ICCICCT Conference, Kumaracoil, India*, pp. 801-804, 2016.

[25] J. Muslimin, A. L. Asnawi, A. F. Ismail, A. Z. Jusoh, N. A. Malek, and H. A. M. Ramli., "Experimental Studies on the Effect of Antenna Orientations to the Performance of OFDM - based System," *International Journal of Electrical and Computer Engineering*, vol. 8(4), pp. 2588-2594, Aug 2018.

[26] GNU Radio is a free and open-source toolkit for software radio. The GNU Radio Foundation, Inc., [Available] http://gnuradio.org/. 2017.

[27] B. Larsson., "Program to decode traffic from Devices that are broadcasting on 433.9 MHz like temperature sensors," [Available] https://github.com/merbanan/rtl_433. 2013.

[28] S. Sanfilippo., "Dump1090 is a simple Mode S decoder for RTLSDR devices," [Available] https://github.com/antirez/dump1090. 2013.

[29] E. Lochin., "STAMP: SMTP Server Topological Analysis by Message Headers Parsing," in *7th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA*, pp. 1-2, 2010.

[30] R. Sureswaran, et al., "Active e-mail system SMTP protocol monitoring algorithm," in *2nd IEEE International Conference on Broadband Network & Multimedia Technology, Beijing, China*, pp. 257-260, 2009.

[31] B. Bencsath and M. A. Ronai, "Empirical analysis of Denial of Service attack against SMTP servers," in *IEEE International Symposium on Collaborative Technologies and Systems, Orlando, FL, USA*, pp. 72-79, 2007.

[32] F. Pereira and L. Gomes, "A JSON/HTTP communication protocol to support the development of distributed cyber-physical systems," in *IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, Portugal*, pp. 23-30, 2018.

[33] K. Munonye and P. Martinek, "Performance Analysis of the Microsoft. Net- and Java-Based Implementation of REST Web Services," in *IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia*, pp. 191-196, 2018.
[34] G. Goyal, K. Singh, and K. R. Ramkumar., "A detailed analysis of data consistency concepts in data exchange formats (JSON & XML)," in *IEEE International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India*, pp. 72-77, 2017.
[35] K. Maeda., "Performance evaluation of object serialization libraries in XML, JSON and binary formats," in *IEEE Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), Bangkok, Thailand*, pp. 177-182, 2012.
[36] S. Cholia, D. Skinner, and J. Boverhof., "NEWT: A RESTful service for building High Performance Computing web applications," *in IEEE Gateway Computing Environments Workshop (GCE), New Orleans, LA, USA*, pp. 1-11, 2010.
[37] S. Ludwig et al., XEP-0166: Jingle. https://xmpp.org/extensions/xep-0166.html. 2016.
[38] S. Ludwig et al., XEP-0167: Jingle RTP Sessions. https://xmpp.org/extensions/xep-0167.html. 2016.
[39] RTP: A Transport Protocol for Real-Time Applications, IETF Standard RFC3550, 2003.
[40] E.Y.T. Adesta, D. Agusman, and Avicenna., "Internet of Things (IoT) in Agriculture Industries," *Indonesian Journal of Electrical Engineering and Informatics*, vol. 5(4), pp. 376-382, Dec 2017.
[41] A.H. Ali, R.F. Chisab, and M.J. Mnati., "A smart monitoring and controlling for agricultural pumps using LoRa IOT technology," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13(1), pp. 286-292, Jan 2019.
[42] R. Gayathri and S. K. Vasudevan, "Internet of Things Based Smart Health Monitoring of Industrial Standard Motors," *Indonesian Journal of Electrical Engineering and Informatics*, vol. 6(4), pp. 361-367, Dec 2018.
[43] S. Jaloudi., "MQTT for IoT-based Applications in Smart Cities," *Palestinian Journal of Technology and Applied Sciences*, vol. 2(1), pp. 9-21, Jan 2019.
[44] A. Bröring et. Al., "The BIG IoT API - Semantically Enabling IoT Interoperability," *IEEE Pervasive Computing*, vol. 17(4), pp. 41-51, Oct-Dec 1 2018.
[45] S. Yang and R. Wei, "Tabdoc Approach: An Information Fusion Method to Implement Semantic Interoperability Between IoT Devices and Users," *IEEE Internet of Things Journal*, vol. 6(2), pp. 1972-1986, Apr 2019.
[46] E.S. Pramukantoro and HusnulAnwari, "An Event-based Middleware for Syntactical Interoperability in Internet of Things," *International Journal of Electrical and Computer Engineering*, vol. 8(5), pp. 3784-3792, Oct 2018.

BIOGRAPHY OF AUTHORS

Samer Jaloudi earned his B.S. degree in electrical engineering from An-Najah National University, West Bank, Palestine, in 1998; his M.S. degree in the design of analog and digital integrated systems from Grenoble Institute of Technology, France, in 2005; and his Ph.D. degree in electrical engineering from the University of Bolton, United Kingdom, in 2013. He is currently an assistant professor with the Department of Information and Communication Technology, Al Quds Open University, Nablus, West Bank, Palestine.
He is the author and coauthor of more than 20 articles. His research interests include information and communication technologies of smart city sectors, integration of renewable energy sources, supervisory control and data acquisition systems, and software-defined radio. He is a senior member of the IEEE.