An Evaluation and Enhancement of a Novel IoT Joining Protocol

Tyler Nicholas Edward Steane
RMIT University

P J Radcliffe
RMIT University

Abstract: The ability to securely join IoT Devices to Wi-Fi networks is an ongoing area of research. This paper describes how Nasrin & Radcliffe's theoretical “novel minimalist IoT network joining protocol” has been mapped to real world hardware and implemented using the Android operating system. For the first time the theory is proven to be practically viable but it is also shown that the user interface is not sufficiently simple for the everyday user. This paper proposes and implements a new user interface paradigm that dramatically simplifies the process and makes the joining process accessible to a much larger range of users. For intensely cost-sensitive applications an alternative process is proposed that has the possibility of even further simplifying the user experience. Finally, the compatibility of the protocol with a variety of operating systems is assessed.

Keywords: Internet of Things, IoT, home automation, smart home, distributed discovery protocol

Introduction

The Internet of Things (IoT) is a field of rapidly growing interest with promising applications in home automation (Gubbi et al , Buyya, Marusic, & Palaniswami, 2013). Solutions are being offered at both a research level and a commercial level. These solutions are encumbered by high expense, complicated interfaces and poor security (Greichen, 1992) and these limitations all have an effect on the user's experience (Chang et al, Dong, & Sun, 2014). Creating a simple and positive user experience is essential to the success of IoT devices, particularly in the area of home automation where everyday consumers want to install and use an IoT device.

One example of a commercially available IoT device in the Home automation space is the ‘Nest’, a smart thermostat for the home with a rotating ring and LCD display for an interface (Nest, 2015). While this is sufficient for simple tasks it is cumbersome for entering the Wi-Fi password. Many devices only need such an interface to connect to a network, thus an interface is often more complex than the other functions of the device and thereby such an interface dramatically inflates the price of the device.

To address these issues a novel network joining protocol specifically for IoT home automation has been proposed in a theoretical fashion by Nasrin and Radcliffe (2016). This work made basic suggestions for the protocol's implementation as well as an interface to accompany its use. This approach would reduce cost and interface complexity by using pre-configured Wi-Fi credentials and the ubiquitous smartphone with its wireless routing capabilities or hotspot.

While this was a promising protocol it had not been physically implemented on existing hardware. This paper details how the protocol has been mapped to the capabilities of the Android OS and then implemented using real Android hardware. The interface will be shown to be insufficient for every day users and so it will also be shown how significant enhancements to the interface have been implemented to improve the user experience.

This paper is organised as follows: Section II reviews existing work in this area. Section III briefly outlines the approach taken by Nasrin and Radcliffe. Section IV details how Nasrin and Radcliffe’s protocol has been mapped and implemented on real hardware. Section V will assess and enhance Nasrin’s interface. Section VI will suggest alternate enhancements that might be explored. Section VII provides an assessment of Nasrin and Radcliffe’s protocol beyond the Android OS. Finally, Section VIII details the areas of focus for future work on this protocol and its implementation.

Existing Work

Four main approaches have been seen in the design of home automation devices, they are: Dedicated IO; Bridge; Central Controller; and a Minimalist approach.

Dedicated I/O

In this approach, additional IO hardware is used solely for the purpose of securely joining the local network. Dedicated IO protocols are often different from the primary communications protocol (usually Wi-Fi); for example, NFC, Bluetooth, or a fully featured user interface (with a screen and keyboard or similar input control) have been included in IoT devices purely for the one-off joining event.

Chen, Pan and Li (2012), for example implemented NFC Tags in devices, but still rely on a traditional network to establish a connection. The need for the user to physically move to each device makes the system, if anything, less usable than a mechanical switch. Others like Piyare and Tazil (2011) have used Bluetooth as a regular communications protocol but only between a phone and central controller. The advantage of this was that end devices didn't need Bluetooth hardware and so are cheaper but comes with the added cost, complexity and inflexibility of a Central controller.

Dedicated IO works well for larger, more complicated appliances where additional IO hardware is trivial or already present. It provides some opportunity for the joining event to be streamlined and the user's experience may be marginally improved. However this approach does not scale well to simple devices where it would greatly increase size, complexity and cost. Compromises made using this approach can result in lower cost but questionable user experiences. The Nest example referred to earlier has a 1.75" round screen with a rotating ring. Rotation of the ring is used to scroll through options and depressing the ring will make a selection. This is an extremely compact approach sufficient for most regular operations but is inconvenient and error prone when used to input a Wi-Fi password to join the device to a network (Nest, 2015). This compromise would reduce the cost of the device compared to a larger keyboard and display but loses the advantage of a positive user experience. The overall cost of the device is much higher than a device with no IO hardware, and the user is still left with a clunky interface.

Bridge

The second approach to solving the network joining problem again uses additional hardware to translate between Wi-Fi and some other protocol, thus creating a bridge. Zigbee has been a popular protocol in this approach (Dou et al , Mei, Yanjuan, & Yan, 2009; Yan & Dan, 2010) as it has secure joining inherent in the protocol (Gomez & Paradells, 2010). Control of a device is thus maintained by a bridging device running a protocol like Zigbee connected to a local router. Other protocols like Insteon and Z-wave have been used; however, like Zigbee they ultimately rely on other protocols and infrastructures to be in place. These secondary protocols are simply bridging the IoT device over to another, primary, protocol to avoid building capabilities into the device to handle the primary protocol. Such an approach only serves to increase cost and complexity of IoT devices and the networks they inhabit (Gomez & Paradells, 2010).

Central Controller

A third approach to network joining uses a central controller to which all devices are physically wired. This physical connection offers an inherent security advantage and reduces the complexity of devices, but it sacrifices the flexibility offered by a wireless connection.

KNX is one of the more mature and successful protocols used in this approach. It is built from several well-established protocols and is tailored to situations such as home automation (Lee & Hong, 2009). Most implementations employ a wired solution, but all approaches rely on a PC or server, as a central controller, being connected and powered on constantly. It also requires a considerable amount of processing power within each device (Zamora-Izquierdo et al., Santa & Gomez-Skarmeta, 2010). This means that as a solution it is very expensive: devices are expensive, central controllers are expensive, running costs are high, and adjustments are difficult and costly.

Such systems are not easily altered and users are quickly locked into a single product line, increasing costs. While the individual devices are simpler, the overall system's complexity is increased and its flexibility reduced, compromising any gains to the user experience.

Minimalist

Finally, Nasrin and Radcliffe's (2016) new minimalist approach has been proposed which uses Wi-Fi and the associated security protocols (e.g. WPA, WPA2) but, unlike the other approaches, removes the need for complex and cumbersome user interfaces. This approach uses the hotspot capabilities of a smartphone to establish an initial temporary, but secure connection to pass credentials for the local home network. This approach requires little or no adjustment to existing controllable devices and is extremely scalable to simpler devices such as mains switches.

The Minimalist approach, with its protocol promising uncompromised security at reduced cost and complexity, is the best approach. However, this approach has yet to be physically implemented; Nasrin has only offered a simple proof of concept which does not automate the hotspot function of the phone and requires attention to improve the user interface. The proposed interface requires a high technical competence from the user and does not streamline the process or maintain a positive user experience. With such modifications, this approach would be of immediate use to the IoT industry.

Nasrin and Radcliffe's Approach

Nasrin and Radcliffe's protocol is designed to connect IoT device to wireless networks while maintaining a high level of security. It is intended to reduce costs by only using the hardware already necessary for a wireless IoT device (Nasrin & Radcliffe, 2016).

Fig. 1 summarises the IoT joining protocol. Each wireless IoT device would be preconfigured with a unique Service Set Identifier (SSID) and password. The device can only be contacted on a network meeting these unique and secret settings. This can most easily be achieved by a smartphone acting in hotspot mode.

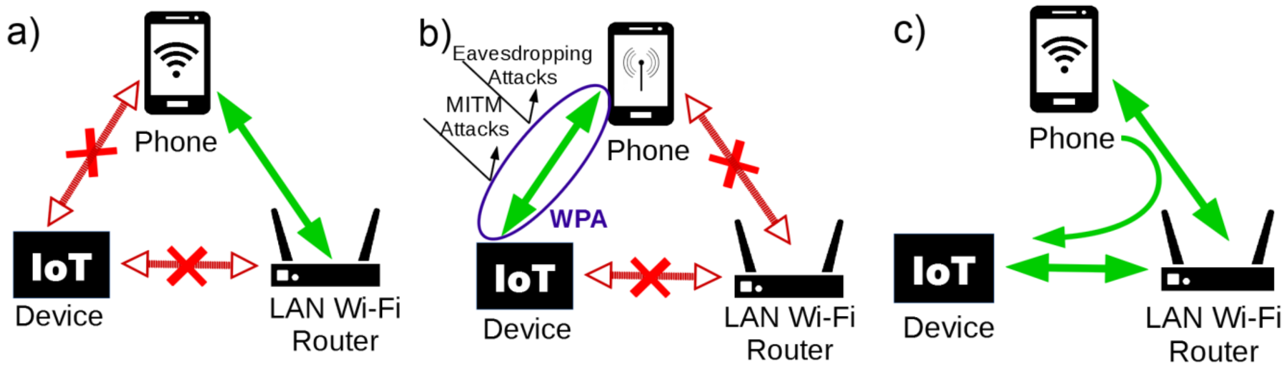

Figure 1: Nasrin and Radcliffe's Steps. a) Initial set-up, b) Hotspot mode transfer LAN details, c) Phone and IoT transfer to LAN

Typically, this temporary link would be secured by WPA2, allowing the SSID and password for the local Wi-Fi network to be passed to the IoT device in a secure manner. Once the IoT device receives the local Wi-Fi credentials it disconnects from the smartphone's hotspot and joins the local network permanently, along with the smartphone. All this is achieved without the need of extra hardware or complexity in the IoT device.

Thus far all that Nasrin has demonstrated is that when a smartphone and an IoT device are connected via the smartphone's hotspot the two can transmit data to one another using the User Datagram Protocol (UDP). Nasrin further envisages a more complete implementation of the proposed joining protocol, by the addition of two features: the first to automate the task of putting the smartphone in and out of Access Point (AP) mode and the second to automate the reconfiguration of the IoT and smartphone Wi-Fi credentials to join to the local Wi-Fi network. These functions need to be implemented to ensure the protocol is practically viable and this is one of the main purposes of this paper.

Mapping & Implementation

In assessing the capacity of current hardware to support Nasrin's protocol, four key functions needed to be achievable. Firstly, it needed to be possible to force a smartphone in and out of AP mode. Secondly, the Wi-Fi configuration of a smartphone needed to be programmatically configurable, in both Wi-Fi and AP mode. Thirdly, communications between an IoT device and a smartphone needed to be achievable. Finally, the Wi-Fi credentials of an IoT device needed to be programmatically configurable.

These key functions were mapped successfully to the Android operating system, while the IoT device was represented by a Raspberry Pi running Linux.

The following assessment was made to verify the compatibility of Nasrin's protocol with current hardware and to identify the tools necessary to implement the protocol:

Smartphone Wi-Fi Programmatic Initiation is possible using setWifiEnabled, a Boolean member of the android.net.wifi.WifiManager API class, which can toggle Wi-Fi on or off (Android Developers, 'WifiManager', 2016c). While the status of the Wi-Fi service can be determined using isWifiEnabled() or for more detail getWiFistate(). The getSystemService method (Android Developers, 'Context', 2016b) will return the WifiManager class to enable the above functions (Mendoza, 2012).

Programmatic Wi-Fi Configuration can be achieved using the addNetwork() member function of the WifiManager Class and configuring the WifiConfiguration class. By editing the WifiConfiguration class and populating the SSID and PreshareKey strings the Wi-Fi can be configured to the desired settings. In order to achieve configuration under AP mode Android 4.0 (API 14) or higher must be used earlier versions do not support configuration (Android Developers, '<uses-sdk>' ,2016a).

Communications were previously developed by Nasrin and Radcliffe (2016) using a UDP link to deliver packets between devices. This communication link is well understood and other possible approaches abound, however the existing work is sufficient.

IoT Wi-Fi Programmatic Configuration under Linux can be achieved by editing the configuration file, wpa_supplicant.conf (Malinen,2013) and then restating the service

All of these functions were successfully implemented and integrated into a functional Android application using the interface proposed by Nasrin and Radcliffe, shown in Fig. 2a-2b). The Raspberry Pi was used to represent an IoT device, in this case a simple Light which can be turned on and off, Fig. 2c). This was done by toggling a TRIAC using the GPIO pins on the Raspberry Pi, and connecting mains and a lamp to the TRIAC.

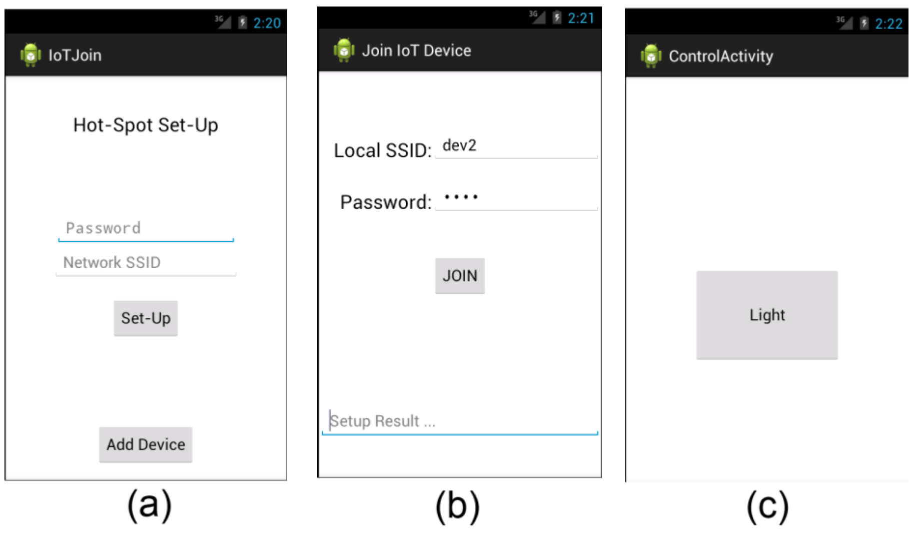

Figure 2 Implementation of Nasrin's interface in Android 4.0

This assessment for the mapping of Nasrin and Radcliffe's protocol has demonstrated the capacity of the Android OS to support the protocol. However further assessment is needed to ensure that other OS are capable of supporting the protocol.

Enhancing Nasrin's Interface

At this point limitations of Nasrin's suggested interface became apparent. It was an effective interface but a rather demanding one, requiring the user to be very familiar with their local Wi-Fi network's configuration. The interface also required typing of random SSID and passwords which increases the likelihood of user error and so creates a poor user experience. The interface further lacked any form of progress indication to reassure the user that the application was working on their requests and not simply waiting for further prompts or had frozen. Key enhancements implemented are discussed below.

Firstly, an automated method for entering the preconfigured credentials of the IoT device was developed using QR codes. Instead of the manufacturer providing a unique written SSID and password they would supply a unique QR code which the user would scan thus saving on typing, errors and time. The successful implementation is shown in Fig. 3.

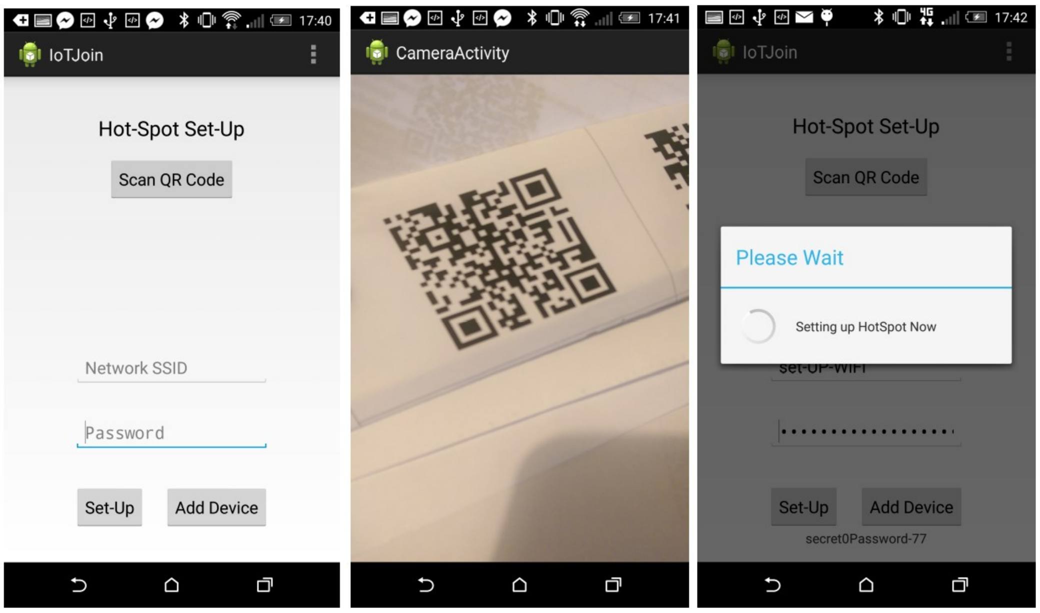

Figure 3 Hotspot setup using QR code scanning

Secondly, users are more accustomed to joining Wi-Fi networks by selecting the desired SSID from a list. This approach has been integrated into the joining application, shown in Fig. 4. Now users only need to be able to identify their SSID (not type it) and enter the password. This provides a familiar process for the user and requires only minimal familiarity with the home Wi-Fi.

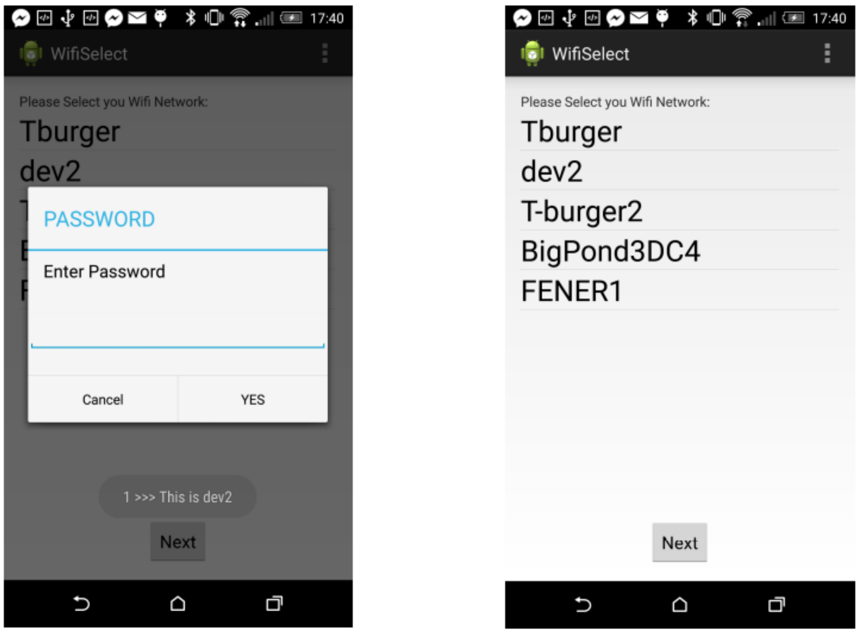

Figure 4 Available Wi-Fi list and joining dialogue

Thirdly, Loading screens are a simple but effective way of reassuring users that the application is working and not simply sitting idle. They can be used to inform of wait times and if something is not configured correctly. If the user's expectations are set early, longer wait times have less impact on the user's experience, whereas unexpected inactivity quickly leave users uncertain and concerned, quickly eroding the user experience.

Fig. 5 shows the complete signalling detail between all actors for the final implementation. The drawing of Fig. 5 caused us to uncover an omitted but important use-case: how can the IoT device be removed from a network and joined to another? One solution is to include a Wi-Fi based command to leave the network but this presupposes that the original network is available to allow this transaction. A better solution can be borrowed from nearly all routers: a very inexpensive pinhole factory reset switch that sets the IoT device back to its unique SSID and password.

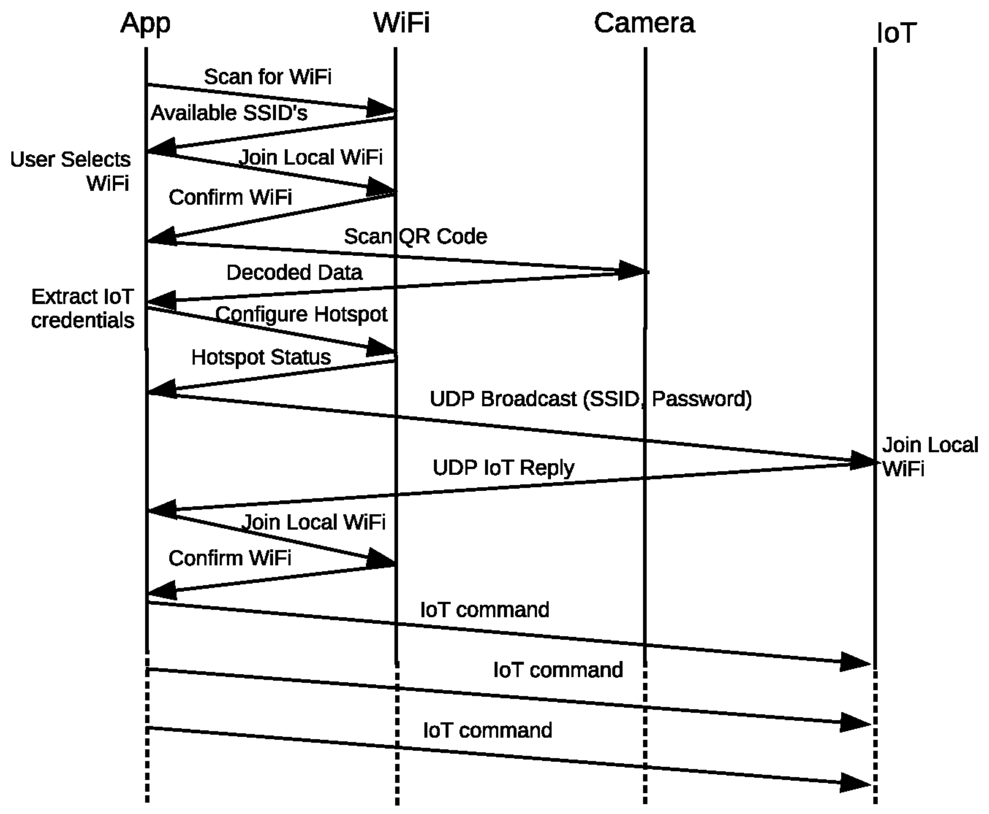

Figure 5 A bounce diagram for a successful connection

Simpler Again & More Secure

Proving that Nasrin & Radcliffe's network joining protocol worked in practice, and making significant enhancements, has helped make the protocol more user-friendly – but is it possible to make the user experience even simpler? While cost has ruled out extra hardware solutions such as adding NFC, these implementations can result in a very simple interface: the user simply selects their own LAN, enters the password, and hits a "join" button.

In order to preserve the minimalist approach of Nasrin and Radcliffe's protocol is it possible to make use of the hardware already inherent in IoT devices to simplify the user's experience? IoT devices often have extra hardware of very low cost, such as an LED indicator or a buzzer, which are under the control of a microprocessor.

It may be possible to use these hardware components to simplify the joining process as much as NFC has achieved. Using Nasrin & Radcliffe's protocol these LEDs or buzzers could securely send the IoT's unique SSID and password to the smartphone thus eliminating the QR code stage. This could reduce costs and simplify the users' experience.

Instead of scanning a QR code users would start up the IoT device which would begin flashing (for the LED) or playing (for the buzzer) a signal encoded with the local SSID and password. Users could then capture this signal from a smartphone application, whether by pointing the camera at the LED or by holding the microphone to the buzzer. This removes the need for QR codes, eliminating the manufacturers cost of generation, printing and packaging. Furthermore, users won't need to keep track of QR codes and which devices they belong to.

This approach adds an extra level of security as the joining device must be present – not just within range of Wi-Fi, but also within range of the buzzer or LED. Thus, the IoT device will be able to verify that it is communicating with a user within a close enough proximity to likely be the owner (or at least possessor) of it.

In Nasrin and Radcliffe's original design of the protocol, if a malicious user happened to know the initial IoT SSID and password and was within Wi-Fi range of the IoT at the time of the devices being powered on prior to joining, they might be able to join the device first or capture the home SSID and password. However, with the use of auxiliary hardware the SSID and password can only be retrieved by a much closer proximity. To prevent malicious users guessing the Wi-Fi credential or predicting them based on manufacturers' patterns, the IoT device could further verify the device either by randomly generating the password or after connecting to the hotspot by requesting a second random pass code which would be encoded on the LED or buzzer. The process by which these passwords are generated need not be overly complex so as to demand significant overheads, and need only avoid basic predictability by avoiding patterns related to the manufacture. The main purpose and value is that it is not a static password.

These options, if feasible, have the potential not only to simplify the users experience but also to reduce the cost of IoT devices as well as to enhance the security of the joining event by verifying the user's proximity to the IoT device.

LED Solution

An LED implementation would require the user to hold the smartphone's camera over a flashing LED to obtain the unique SSID and password for the IoT device.

An IoT device can modulate an LED at high speed but a smartphone can only monitor such an LED using video capture at frame rates as low as 12 Hz up to 60 Hz or more depending on the smartphone. Using the Nyquist sampling criteria this means the data rate at best will be between 6 Hz and 30Hz. While an SSID can be quite short, perhaps only 2 letters, a password should be much longer. For WPA2 it has been suggested that a password be a minimum of 12 random characters which equates to 78 bits of entropy (Farik & Ali, 2015a; 2015b). Thus, a minimum of 14 characters, 2 for SSID and 12 for a password, each of 7 bits is required; 98 bits. At a 6 Hz sample rate a user would thus have to hold the camera over the

IoT LED for approximately 17 seconds, or 4 seconds for a high-speed smartphone. These times are based on calculations for a baseband signal, they may be reduced by more complex modulation and encoding but this would come at the cost of increased computational overheads which may not be viable for constrained simple devices. It would be quite a challenge to design a user interface that would help a user cope with these time lengths. Scanning a QR code would appear to be a much simpler solution for the user.

Buzzer Solution

The majority of buzzers are of a self-resonant type which buzz at a predetermined frequency when supplied with a DC voltage. In terms of modulation this implies that amplitude modulation (AM) is the only viable modulation technique. When a small buzzer is turned on and off the attack and decay times are typically 2-3 milliseconds which suggest a data rate of at least 100 Hz is possible. A smartphone can capture and record audio information at speeds from 3 kHz up to 44 kHz depending on the quality of the smartphone. Assuming a low data rate of only 100 bits per second this means that the smartphone could capture the IoT SSID and password in around one second given that AM modulation is the only viable modulation method.

While this solution seems viable the annoyance value of a buzzer must be taken into consideration. A continually operating buzzer is likely to annoy a user and erode the user experience significantly. Perhaps the factory reset switch previously mentioned could be used to play out the SSID and password several times and then stop.

Compatibility

Having now implemented Nasrin and Radcliffe's protocol we are in a position to accurately understand the requirements of the protocol and to consider the compatibility of the protocol with other operating systems.

Under Android, API's are available to grant access to control the Wi-Fi Hardware and as such the application manifest must declare that this application will be granted access to these settings. Can other mobile operating systems grant such access or is such access blocked? Most notably, Apple's iOS does not allow developers to programmatically configure the Hotspot mode, nor even to switch it on or off; some access is available to Wi-Fi connectivity and configuration but it is very limited and tightly controlled (Apple Developer Documentation, "CoreWLAN", 2017). While this has been confirmed by a careful assessment of the developer's documentation, it is also widely confirmed by the official and unofficial developer forums.

While Android has a large share of the market at 86.8% globally it is by no means without competition (IDC, 2017). However, in countries like Australia Android leads but only with 52.3% of the market followed closely by Apple with 44.9% (Kantar Worldpanel ComTech, 2017). Similarly, in Great Britain Android has 50.6% of the market share compared to 47.6% held by Apple and in the USA Android holds 54.4% with Apple at 44.4% (Kantar Worldpanel ComTech, 2017). Given these market figures, any protocol should work with both Android and iOS.

Without the ability to programmatically interface with the Wi-Fi hardware under iOS, a smooth user experience is not possible. However, it would not even be possible to implement Nasrin and Radcliffe's protocol with manual Wi-Fi configuration. This is because users cannot configure the Hotspot SSID, which is created based on the devices name.

Furthermore, not all Android devices are equipped with hotspot capabilities or they have been disabled. It is therefore, on the whole, not reasonable to assume that anyone wanting to configure a home automation system will have access to compatible hardware. Thus, it would seem that while Nasrin and Radcliffe's Protocol can be practically realised and with some enhancements it can be made user-friendly, it ultimately fails to address the availability of hardware to the average user.

The best solution would be a universal joining protocol compatible with any devices that supports Wi-Fi. Currently Nasrin and Radcliffe's protocol is not able to support this and so further work is needed to make it a viable solution for all users in the DIY home automation market.

Future Work

The enhanced version of Nasrin and Radcliffe's protocol is very viable but the use of an LED or buzzer to send the IoT SSID and password to the smartphone does have the potential to further simplify the user's experience. Developing an LED-based solution is a real challenge requiring an outstanding user interface or some very novel sampling methods. The buzzer solution appears more viable and is worth further investigation.

Nasrin and Radcliffe's protocol has been successfully implemented for the first time, and this has allowed an assessment of its compatibility with common devices. This assessment raises concerns around the limited compatibility of the protocol as it is essentially only compatible with Android devices with hotspot capabilities. Further work will investigate alterations to the protocol to widen its range of compatibility to include at least iOS. A joining protocol for home automation devices must be as widely compatible as is reasonably possible.

Conclusion

It has been demonstrated by a consideration of the literature that, in theory, Nasrin and Radcliffe's protocol is both a novel and superior network joining protocol for IoT devices. Careful analysis has shown that the protocol can be mapped to the capabilities of the Android operating system. The physical implementation of Nasrin and Radcliffe's protocol, using an Android smartphone and a Raspberry Pi, has proven that the protocol is practically viable. It was discovered that while Nasrin's user interface was functional it was not friendly for the everyday user. This paper proposed and implemented three new user interface features that made the network joining process notably easier for the average user.

Many simple IoT devices have an LED or buzzer under control of a microprocessor. This paper has analysed the potential for these devices to replace the QR code portion of the joining protocol, and concluded that such an approach is viable and worthy of future research.

Finally, the compatibility of Nasrin and Radcliffe's protocol has been determined to be limited to Android devices with hotspot capabilities. While this limitation still includes many devices, it is still incompatible with many popular devices notably those running iOS. This provides an additional area for further research.

Acknowledgements

Thanks to Kai Xu for his work testing LED’s and Buzzers.

This research was supported by an Australian Government Research Training Program Scholarship.

References

Android Developers. (2016a). <uses-sdk>. Retrieved June 13, 2016, from https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#uses

Android Developers. (2016b). Context. Retrieved June 13, 2016, from https://developer.android.com/reference/android/content/Context.html

Android Developers. (2016c). WifiManager. Retrieved June 13, 2016, from https://developer.android.com/reference/android/net/wifi/WifiManager.html

Apple Developer Documentation. (2017). CoreWLAN. Retrieved March 17, 2017, from https://developer.apple.com/reference/corewlan

Chang, Y; Dong, X; Sun, W. (2014). Influence of characteristics of the Internet of Things on consumer purchase intention. *Social Behavior and Personality*, *42*(2), 321–330. Available at https://doi.org/10.2224/sbp.2014.42.2.321

Chen, L; Pan, G; Li, S. (2012). Touch-driven interaction via an NFC-enabled smartphone. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)* (pp. 504–506). Available at https://doi.org/10.1109/PerComW.2012.6197548

Dou, N; Mei, Y; Yanjuan, Z; Yan, Z. (2009). The Networking Technology within Smart Home System - ZigBee Technology. In *International Forum on Computer Science-Technology and Applications, 2009. IFCSTA '09* (Vol. 2, pp. 29–33). Available at https://doi.org/10.1109/IFCSTA.2009.129

Farik, M; Ali, A. S. (2015a). Algorithm To Ensure And Enforce Brute-Force Attack-Resilient Password In Routers. *International Journal of Technology Enhancements and Emerging Engineering Research*, *4*(10), 184–188.

Farik, M; Ali, A. S. (2015b). Analysis Of Default Passwords In Routers Against Brute-Force Attack. *International Journal of Technology Enhancements and Emerging Engineering Research*, *4*(9), 341–345.

Gomez, C; Paradells, J. (2010). Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, *48*(6), 92–101. Available at https://doi.org/10.1109/MCOM.2010.5473869

Greichen, J. J. (1992). Value based home automation for todays' market. *IEEE Transactions on Consumer Electronics*, *38*(3), XXXIV–XXXVIII. Available at https://doi.org/10.1109/30.156666

Gubbi, J; Buyya, R; Marusic, S; Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660. Available at https://doi.org/10.1016/j.future.2013.01.010

IDC. (2017). IDC: Smartphone OS Market Share. (2017). Retrieved February 24, 2017, from http://www.idc.com/promo/smartphone-market-share/os

Kantar Worldpanel ComTech. (2017). Smartphone OS sales market share. Retrieved February 24, 2017, from https://www.kantarworldpanel.com/smartphone-os-market-share/

Lee, W. S; Hong, S. H. (2009). Implementation of a KNX-ZigBee gateway for home automation. In *2009 IEEE 13th International Symposium on Consumer Electronics* (pp. 545–549). Available at https://doi.org/10.1109/ISCE.2009.5156866

Malinen, J. (2013) Linux WPA Supplicant (IEEE 802.1X, WPA, WPA2, RSN, IEEE 802.11i). Retrieved June 13, 2016, from http://w1.fi/wpa_supplicant/

Mendoza, A. J. (2012). Tutorial For Android: Turn off, Turn on wifi in android using code tutorial. Retrieved June 13, 2016 , from http://www.tutorialforandroid.com/2009/10/turn-off-turn-on-wifi-in-android-using.html

Nasrin, S; Radcliffe, P. J. (2016*). A Novel Three Stage Network Joining Protocol for Internet of Things based Home Automation Systems. Computer Communication & Collaboration, 4(3)*, (pp. 1-11).

Nest. (2015). Nest Protect and Nest Cam support. (2015). Retrieved June 13, 2016, from https://nest.com/support/article/A-step-by-step-guide-to-setup-on-the-Nest-Learning-Thermostat

Piyare, R; Tazil, M. (2011). Bluetooth based home automation system using cell phone. In *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)* (pp. 192–195). Available at https://doi.org/10.1109/ISCE.2011.5973811

Yan, D; Dan, Z. (2010). ZigBee-based Smart Home system design. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)* (Vol. 2, pp. V2–650–V2–653). Available at https://doi.org/10.1109/ICACTE.2010.5579732

Zamora-Izquierdo, M. A; Santa, J; Gomez-Skarmeta, A. F. (2010). An Integral and Networked Home Automation Solution for Indoor Ambient Intelligence. *IEEE Pervasive Computing*, *9*(4), 66–77. Available at https://doi.org/10.1109/MPRV.2010.20