computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

IOT STATUS COMMUNICATION FOR HOME AUTOMATION

Tyler Nicholas Edward Steane, PJ Radcliffe

School of Engineering, RMIT University, Melbourne, Australia
tyler.steane@rmit.edu.au, pj.radcliffe@rmit.edu.au

Paper history:
Received 4 March 2019
Received in revised form 28 August 2019
Accepted 10 September 2019
Available online 30 September 2019

Keywords:
Home Automation;
IoT;
M2M;
Network protocol.

Abstract: The vast majority of home automation systems depend on a permanent central controller and this is the source of many serious problems. The controller may be eliminated by using one or more Intermittent Control Devices such as a smartphone, but the responsibilities of the controller must be redistributed. This paper considers the problem of keeping multiple intermittent controllers up-to-date with status information for devices around the home. Four protocols are proposed, taking key lessons from existing network protocols, and then implemented, tested and compared with the expected performance of UPnP. All four protocols outperform UPnP and further analysis shows that these protocols can be implemented in a robust and user-friendly manner. A comparison of the packet efficiency of these protocols demonstrates that the combination of device registration and packet broadcasting makes for the most efficient protocol and should form the basis of devices status communications in home automation systems without a permanent central controller.

1. INTRODUCTION

Home automation and smart homes have long been considered an imminent and inevitable application of technological advances in microprocessors, sensors and networking. Despite this such systems are still far from the ubiquitous technology anticipated. It has been argued this is largely the result of issues either introduced or exasperated by the near universal dependence of Home Automation systems on a Permanent Central Controller (PCC) [1]. These PCC's are expensive to purchase and run, creating an obstacle for the average household. Furthermore, they are often a wired solution requiring professional installation and maintenance. Add to this such systems are complex to re-configure leaving most users unable to make even minor changes to the system. Complex and inflexible systems will not succeed in penetrating homes where users are not technically confident.

Today's market is growing with individual standalone automated devices, from light globes to TVs but each with their own specialized app, this prevents any form of an integrated system with co-operating devices being achieved. Additionally, users are then burdened with navigating many disparate applications on the one smartphone to achieve basic control functions. This approach will not suffice as a means to eliminate the PCC, as it sacrifices the interoperability of an integrated system and burdens the users with a cumbersome interface.

To address the issues of expensive, complex, inflexible and proprietary Home Automation Systems it is none the less essential that the PCC be eliminated. However, many tasks and responsibilities are associated with the PCC and this poses a significant challenge for successfully eliminating the controller, without making the compromises of standalone devices. The dominant system topology found today revolves around remote interfaces (e.g. wall panels, remote controls, smartphones and Web pages), accessing devices via the PCC. Our solution has been to promote remote interfaces, such as smartphones, to the role of Intermittent Control Devices (ICDs) and to require compliance to simple protocols by smart devices in the home, see Fig. 1, thus redistributing the responsibilities of the PCC and maintaining overall functionality. To date we have demonstrated that this approach can successfully and securely join IoT devices to a home Wi-Fi network [1] and then discover those devices [2].

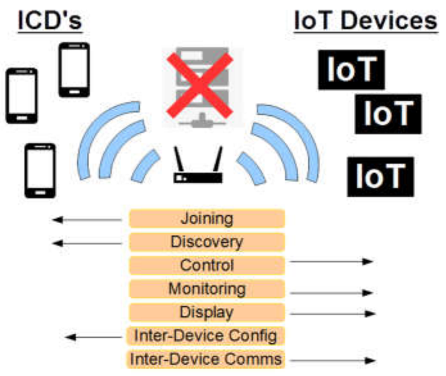

Figure 1 – Redistribution of responsibilities of a Permanent Central Controller

The next challenge is the cluster of tasks and responsibilities servicing the display and control of devices. The aim of these tasks is to allow ICD's to generate a UI prescribed by the IoT device as well as control and monitor its status. This paper addresses the issue of multiple ICD's in one system and the challenge of keeping all their status displays appropriately up-to-date, monitoring in Fig. 1.

Traditionally with the use of a PCC, users' devices would go to the controller directly for the status of any given device. The PCC was the source of 'truth' for all devices on the system and would have the most up-to-date data on a devices status. The controller would also manage the update intervals for each device and its various status values. The question is now, how will an ICD ensure it has status data that is sufficiently up-to-date for each specific device and its application? And as an additional complication how will multiple ICDs maintain up-to-date status' not only of the device's private values but also those which may be adjusted by another ICD.

Any solution to this problem must minimize network traffic as the home environment may quickly become very busy as more and more IoT devices are added to the home automation system. One network could easily consist of 150 or more devices and so a successful solution will minimize packets even as it maintains a timely update schedule for multiple ICDs.

In this paper four novel protocols are proposed to allow multiple ICDs to maintain up-to-date status information from IoT Devices. These protocols are assessed and compared, against one another and UPnP, for packet efficiency and provide a viable solution with minimal traffic and therefore little disruption to the home network. Further analysis considers the measures to be taken to ensure a robust and user-friendly implementation of the protocols.

This paper is organized as follows: Section 2 will present a consideration of existing work and how it can benefit the problem considered by this paper. Section 3 will present the design of the proposed protocols for developing a solution, while section 4 will present the methodology used to assess the proposed solutions followed by a discussion of these results in section 5. Section 7 will provide further analysis of the protocols robustness and usability. Finally, section VI considers where further work could focus. This work extends previously published work [3].

2. RELATED WORK

Home automation has long been the realm of proprietary protocols, but more recently open standards have been gaining significant momentum. The two major players to consider are MQTT [4], first standardized in 2014, and CoAP with a draft RFC published in 2014 [5].

The Message Queuing Telemetry Transport (MQTT) protocol is a lightweight protocol for machine-to-machine (M2M) communications in the Internet of things. It has been implemented in a variety of systems targeting home automation. Largely due to its limited computational demands several applications have shown that it can be implemented on low cost hardware such as Raspberry Pi's [6, 7], Arduino boards [6, 8] and ESP8266 based boards [8–10]. While all these systems depend on the ubiquitous home Wireless network some research has considered the use of other physical layer protocols such as LoRa to carry the MQTT communications [11]. While LoRa has many promising applications the added complexity and cost of another physical protocol's demand for additional hardware in the home will prove a great obstacle for many and its long-range advantage has limited application in the home.

While the MQTT base Home Automation systems propose low cost hardware they are not part of a wider protocol that considers device joining and discovery and they are bound to a Central Controller. Jutadhamakorn et al. [7] demonstrated that this approach can be made to scale reasonably well but this is only achieved by further embracing the complexities of a Central Controller to the neglect of a simple user experience. This dependency is not simply a design choice as MQTT is fundamentally based on a central controller in the form of what it calls Brokers.

Brokers are the middle-man in MQTT's three component paradigm of Publisher, Broker and Subscriber [4, 12]. The Broker is a subscriber's source for information supplied to the Broker by a

Publisher. Publishers send the Broker data associated to topics which subscribers can register their interest for or subscribe to. While this is a good approach for reducing network traffic and still keeping interested parties up-to-date, it is entirely and fundamentally dependent on a central controller.

The Constrained Application Protocol (CoAP) is a more recent protocol that targets low power devices with limited computational capacity. Though less work has considered the application of CoAP to Home Automation Systems, its fundamental lightweight philosophy is helpful. CoAP has been used as a communication layer to achieve interoperability between different, often legacy protocols [13, 14]. Other systems have focused on the security of devices or using CoAP for energy management [15–17] within the home. While all these systems utilize Wi-Fi and sometime Ethernet, Son et al have demonstrated CoAP over other ‘non-IP’ protocols such as Zigbee, UART and RS232 [18].

Once again, all these systems are fundamentally reliant on a central controller. While CoAP may be able to operate under limited circumstances without a central controller, it is designed to perform in a wide system with much more infrastructure than the average home, thus some of its more efficient features are lost, such as the use of intermediaries sharing cached data with multiple interested clients [19].

Like MQTT, CoAP also has a subscription model where clients can register interest in a particular status or ‘resource’. This process is known as ‘observing’ under CoAP and helps reduce packets by registering a client’s interest and keeping them up-to-date without prompting [19].

Some comparison has been made of MQTT and CoAP in home settings, finding CoAP to be more packet efficient but less power efficient than MQTT [20]. However, the main concern for this paper will remain packet efficiency and a ‘Permanent Central Controller’-free paradigm.

UPnP is a much older standard [21], standardized in 2008 and offers a paradigm not fully reliant on PCC. UPnP uses “Eventing” to publish status updates to subscribers when a status variable changes. These Event notifications can be sent as either uni- or broad-casts. This approach is not suitable for variables that may change frequently as this would generate greater network traffic. UPnP also leaves subscribers to make their own assumptions as to whether no updates means the publisher is now offline or just unchanged. Therefore, UPnP Eventing isn’t well suited to the home environment, so under UPnP variables would have to be polled manually using request commands.

3. PROTOCOL DESIGN

To provide a robust and sensible solution to maintain the status data of devices across ICD’s the problem is first simplified by limiting the operation of the protocol to the time when ICD’s are observing a particular devices status, that is active. This limits the challenge to a protocol that can manage the device updates and set their frequency. To this end four potential solutions are proposed, and their behavior investigated. The main discriminating factor between these protocols will be the number of packets generated by the protocol as the home wireless environment can already be a busy one and should not be loaded unnecessarily. All solutions will use simple UDP packets for communication. These protocols aim to take advantage of key insight from MQTT and CoAP, namely registration of interest and the efficiency of intermediaries. These will be compared with the theoretical performance of UPnP using a simple polling command issued at the ICD’s desired update interval, these are calculated since the protocol is so simple.

The first protocol (Fig. 2) is the simplest and is meant to operate as a benchmark as it minimizes packet transmissions while sacrificing flexibility. In this protocol the IoT devices to be observed will continually broadcast their status to the network at a fix interval and interested ICD’s can listen in and record the devices status'. However, since the IoT device will continue to broadcast its status even if no devices are interested it is not a practical solution but is meant only as a comparative bench mark.

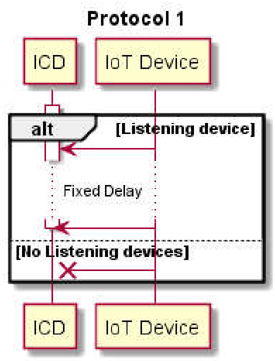

Figure 2 – Sequence diagram for Protocol 1

The second protocol (Fig. 3) makes ICD’s a more active part of the solution, allowing the IoT devices to only transmit when an ICD is interested. ICD’s, therefore, must first register their individual interest with the IoT device along with the interval with which they should be updated, much like the subscription or observing of MQTT and CoAP. Additionally, this registration would constitute an ongoing interest until the ICD announces it is no longer interested. IoT devices will then transmit directly to ICD's at their appointed intervals until the ICD announces it is no longer interested.

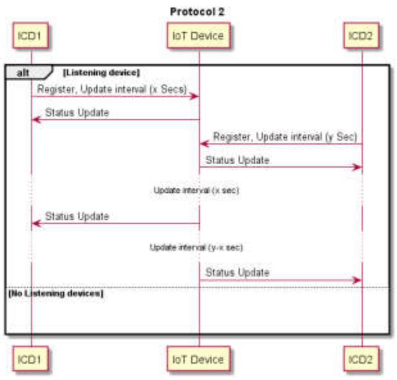

Figure 3 – Sequence diagram for Protocol 2

The third protocol (Fig. 4) reduces the load placed on the IoT device by the previous protocol by having ICD's make a more general registration of interest, simply notifying the IoT device that "someone" is interested and how often they require updating. The IoT device then only keeps track of how many ICD's are interested and the shortest update interval, and then broadcasts at that interval until the number of interested ICDs reaches zero.

While again adopting a registration model from MQTT and CoAP, this protocol also builds somewhat off the idea of intermediary caches in CoAP. This approach reduces the number of replies sent by an IoT device, but instead of using intermediaries it sends a broadcast packet to all ICD's who can keep the data if interested.

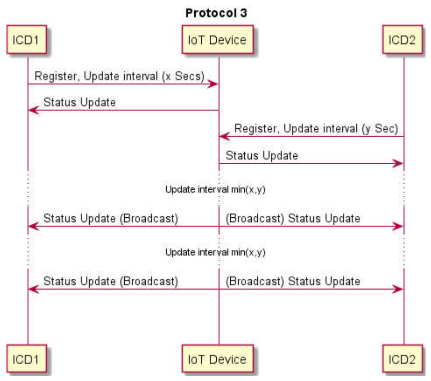
Protocol 3
ICD1
IoT Device
ICD2
Register, Update interval (x Secs)
Status Update
Register, Update interval (y Sec)
Status Update
Update interval min(x,y)
Status Update (Broadcast)
(Broadcast) Status Update
Update interval min(x,y)
Status Update (Broadcast)
(Broadcast) Status Update
ICD1
IoT Device
ICD2

Figure 4 – Sequence diagram for Protocol 3.

The fourth protocol (Fig. 5) minimizes the computational demand on the IoT device by requiring it only to listen and reply to update requests rather than also keeping track of interested devices. ICD's still make individual requests for updates at regular intervals and the IoT device broadcast its response. However, when an ICD receives an update, that it did not request, it accepts the update and resets its interval timer and increases the interval slightly by a fixed back-off interval to allow time for another device to request its next update before the timer expires and it requests its own update.

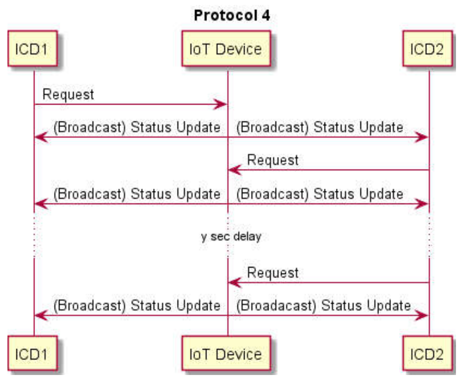
Protocol 4
ICD1
IoT Device
ICD2
Request
(Broadcast) Status Update
(Broadcast) Status Update
Request
(Broadcast) Status Update
(Broadcast) Status Update
y sec delay
Request
(Broadcast) Status Update
(Broadacast) Status Update
ICD1
IoT Device
ICD2

Figure 5 – Sequence diagram for Protocol 4

While this method retains the efficiency of broadcast packets, prompted by CoAP's intermediaries, it is isolated from the registration process which is not integrated in this approach.

4. METHODOLOGY

Four candidate protocols have been proposed for updating ICD's of an IoT device's status. In order to assess their performance each protocol was implemented in C++ scripts for both IoT devices and ICD's. The test-bed architecture developed was very effective and may well be of use to other protocol analysis.

Assessing a network protocol can be achieved in several ways: analytical modeling, simulation, emulation and implementation. Analytical modeling can provide key insights into performance, such as in [22] and [23] where it was used to assess novel approaches to network protocols like CSMA/CA and to assess the reliability of systems utilizing service-oriented architectures. However, the protocols considered in this paper can involve 5 or more complex finite state machines which would create excess complexity in an analytical model and such a model may still miss important real world phenomena. Neither simulation nor analysis can easily offer the same insight into the behavior of a protocol as when implemented on a real TCP/IP stack. Emulation matches all the real-world rules of implementation but may use different hardware and software to decrease development effort. For this reason the protocols were emulated using Virtual machine (VM) instances of Tiny Core Linux 9.0 [24] running the C++ scripts for each protocol [25]. These scripts included diagnostic code to record packet arrival times as well as the total number of packets sent.

Six VM's where used in assessing each protocol, one to represent an IoT device, four representing ICD's and one used to co-ordinate all experiments. These VM's where managed and networked using GNS3 v2.1.8 [26] which imported the Tiny Core VM's from Virtual Box v5.2.12 and provided the Ethernet switch for connectivity. Initial testing is using a wired network free from the confounding effects of wireless networks, but future work will need to consider these effects. Each of the VM's where given a static IP addresses and ran an FTP server to simplify file transfers to the virtual network and ran compiletc v0.1 to compile the C++ scripts.

Each of the four protocols were run for 300 seconds with only 1 ICD active (that is observing) and then for 2 ICDs and so one until the final test with all 4 ICDs active. This suite of tests was repeated 3 times, each with a different combination of update intervals set in the ICD's. The default interval was set at 5 seconds active in the first round of tests all ICD's were set to request the default interval. The second round set the first ICD update to 10 seconds and in the third round it was set to 1 second.

Prot. 2-4 have a set-up time and they are not operating normally for the first few seconds of the test. All tests were run for 300 seconds to minimize the impact of the setup times.

Rather than manually configuring the network for each experiment, which would require setting & configuring the number of active devices and their individual update intervals, the co-ordinator machine was used to automate all the experiments for each protocol. After each experiment the co-ordinator extracted the number of packets transmitted by each device and recorded the total. This meant that all the experiments could be completed by running 4 batch experiments, one for each protocol. This experiment test-bed was very successful. It proved to be very effective and flexible, it is highly adaptable and could easily be applied to many other investigations of protocol performance.

5. RESULTS AND DISCUSSION

The first suite of tests had all devices expecting updates every 5 seconds. The results of the 300 second tests are shown in Fig. 6. An initial assessment of the results would favor the impractical Prot. 1 as the best performer but the close performance of Prot. 3 should not be overlooked. The fact that Prot. 1 will maintain its transmission rate even when zero ICD's are active, sees the performance of Prot. 3 as clearly superior given its marginal increase in packet of Prot. 1.

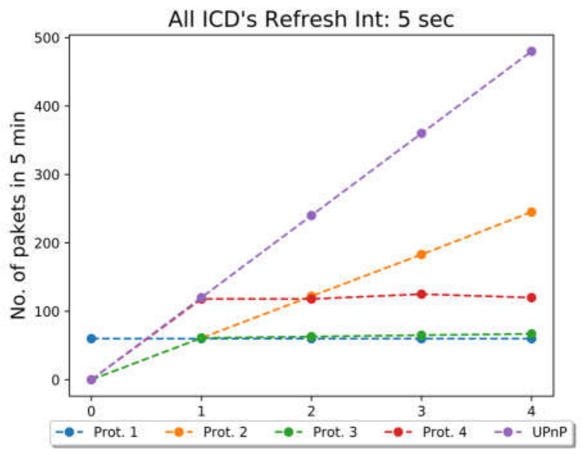

Figure 6 – No. of packet sent in 5mins with 1-4 ICDs active all with an update interval of 5 sec.

Prot. 4 reduces the demands placed on the IoT device but it's behavior is roughly twice as busy as Prot. 1 or 3. Finally Prot. 2 is problematic as it is extremely sensitive to the number of active ICD's.

In the second suit of tests the first ICD to be activated was set to require a longer update interval of 10 sec, while all other ICDs remained at 5 sec. The results can be seen in Fig. 7, the impractical Prot. 1 has the highest number of packets transmitted for 1 active ICD. This highlights the weakness of the protocol in it inflexibility, since all the other protocols take into consideration the desired update interval of the ICD even Prot. 4 outperforms the impractical Prot. 1 in this test. After that, however, the addition of the 2nd, 3rd & 4th ICD, all with the same update intervals as the IoT device sees Prot. 1 perform best, but closely followed by the more practical Prot. 3 (as noted in the previous suite of tests).

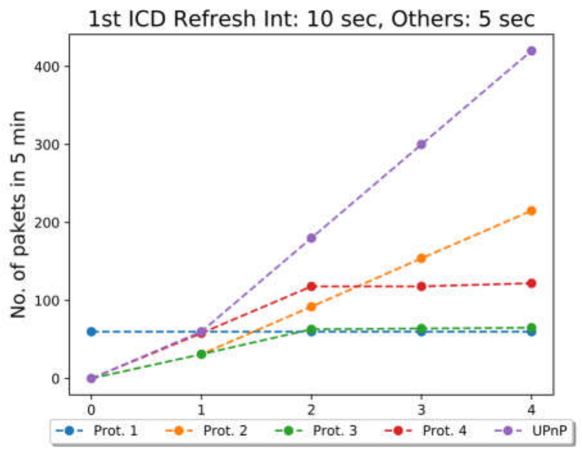

Figure 7 – No. of packet sent in 5mins with 1-4 ICDs active, with an update interval of 10 sec for the first ICD and 5 sec for all others

Prot. 2 follows the same linear behavior as seen in Fig. 6 unaffected by the change in refresh rates except for the total number of transmissions. Prot. 3 & 4 however, present as more affect by the longer update interval demonstrating that they are more sensitive to the update interval than the number of active ICDs.

Fig. 8 shows the packet rates over the 300 second tests this time with the first ICD configured for the shorter update interval of 1 sec. Here again the impractical Prot. 1 shows misleadingly good results due to its inflexible approach to update intervals.

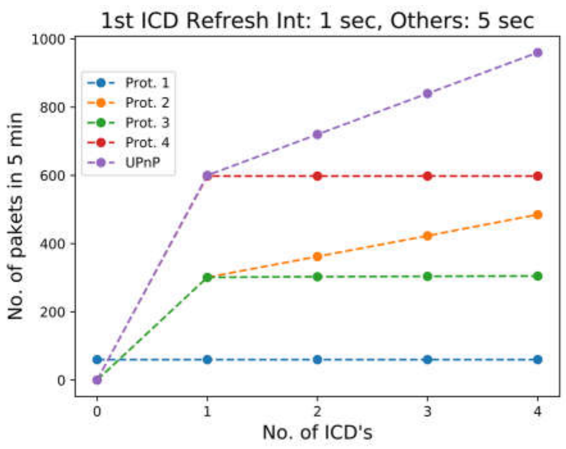

Figure 8 – No. of packet sent in 5mins with 1-4 ICDs active, with an update interval of 1 sec for the first ICD and 5 sec for all others

As previously, Prot. 3 & 4 are both shown to cope well with higher numbers of ICD's especially when they have longer update intervals, due in large part to use of broadcast packets.

The higher packet rate from Prot. 4 highlights the price paid for a reduced load on the IoT device, a price that sees it outperformed by the previously much weaker Prot. 2, which benefits from the registration process even if it is required to address each ICD individually. Prot. 4 has a much more stable packet count, consistently 598 packets. This is due to the significantly shorter update interval of the first ICD, making it the clear 'leader' when initiating broadcast updates from the IoT device. When there are multiple ICD's with update intervals at or near the shortest, as in Fig. 6 & 7, a significant number of extra packet can be generated in the contention phase as ICDs sort out which devices will be the leader requesting updates and which will just listen to the broadcast responses. This contention process may also be re-ignited if the leader's requests ever lags behind ICDs that join after the leader. For this reason, the back-off interval is used to increase the update interval of devices when they recognize they are not in the lead.

The results shown here have minimal contention due to the use of a 1.5 second back-off interval. A lower interval of 1 second resulted in far less consistent behavior with more contention events and higher packet counts.

These results compare well with the predicted performance of UPnP using polling commands. All four protocols out-perform UPnP despite UPnP responding better than Prot. 1 to changes in update intervals. While UPnP closely matches Prot. 4 for only a few ICD's it does not scale nearly as well.

Overall Prot. 3 comes out as the most packet efficient protocol, thanks to the combination of registered interest and broadcast updates. It strikes a balance between the inflexible Prot. 1 and the simple computation (for the IoT) yet talkative Prot. 4. Prot. 3 is much simpler than Prot. 2 and yet yields a much more efficient use of packets.

However, in less constrained environments, where back losses can be handled more easily or are less likely, the reduced computational load on the IoT device may make Prot. 4 more attractive or even Prot. 1 if the conditions are right.

6. FURTHER ANALYSIS

While the protocols have been shown to be variously packet efficient the robustness and usability of the protocols deserves further consideration.

6.1 ROBUSTNESS

In order to analyze the robustness of the protocols potential problematic cases are considered that may arise in the regular use of these protocols. The second and third protocols require ICDs to register their interest and then announce when they are no longer interested in the IoT devices status. However, if devices fail to announce this or the announcement is somehow missed by the IoT device then the packet efficiency of the protocol would be adversely affected. For example, an ICD may simply and easily go out of range (leave the house) of the wireless network and not be able to announce its lost interest in the IoT device. A lease mechanism should be added to these protocols as a catch all to prevent Prot. 2 and 3 acting more like Prot. 1 and constantly broadcasting their status to a network of uninterested devices. The ICD's themselves should be able to request this lease period as various applications may have different demands, and these protocols may be useful for IoT to IoT device communications such as thermostat to Heater or light sensor to curtain-controllers. It may still be possible for packets to be broadcast without an interested device present if leases are too long and ICDs fail to communicate their changed interest. To avoid this problem IoT devices should limit the acceptable lease times and either refuse request that are too long or simply apply the set maximum allowed lease.

Another robustness concern is the ability of ICDs to request an update interval from the IoT devices. Very short updates periods could produce significant traffic possibly to the level of a Denial of Service attack. Once again, the maximum update frequency should be limited by the IoT devices. A high update frequency from an ICD would be reduced to this limit. With IoT devices limiting lease periods and update intervals it may be necessary in certain cases for the ICD to know what period and interval has been granted. The update request could have a field indicating if this service is required.

The robustness considerations have added to the registration packet which now becomes-

1. A particular ICD is interested in updates,
2. the requested lease period,
3. the requested update interval, and
4. whether a reply is required.

6.2 UI CONSIDERATIONS

The primary use for these protocols is for ICD's to provide up-to-date status information from IoT devices to users. While most TCP/IP base protocols are highly responsive with minimal delays it is valuable to consider the timing performance of the protocol to ensure the behavior of the protocols will not impact on the users' experience. Galitz has offered many worthy considerations on how UIs can be designed and implemented to maximize users' experiences [27]. The main contribution of interest here is that users prefer consistency in delay times over shorter delays. Galitz offers the general rule that the deviation of delays should not exceed half the mean:

$$\sigma < \frac{mean}{2}, \quad (1)$$

Delays in updating UI elements that do not obey this rule can be the source of concern or anxiety in users or can cause users to neglect the element altogether. The protocol used to update ICDs should allow the ICD to maintain refresh rates that meet Galitz's rule. To assess the ability of Prot. 2-4 to support this rule the delay of each update packet was recorded against its expected arrival time. Some of the protocols, like 2 and 4, allow for early packets to arrive so these packets would have a negative delay but were truncated and recoded a zero since early data can be buffered.

Packet delays were recoded with four interested ICD's all being updated at 5 second intervals, essentially the last data points of Fig. 6. The distribution of packet delays for the ICD with the highest ratio of standard deviation to mean for each protocol are shown in Fig. 9.

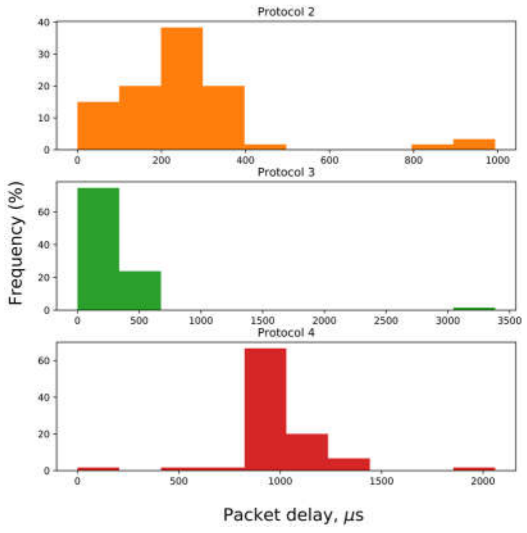

Figure 9 – Distribution of packet delays for the device with the largest ratio of STD to mean

Comparisons between the different protocols is of limited value since the sample sizes are very different due to the different packet rates of each protocol. However, they are all reasonably close ranging from ~3000 to 0 μs, with negative delays truncated to 0s. The data in Fig. 9 records how much the packets where delayed but not the amount of time between packets. To get the time between packets 5 seconds must be added to the delay of each packet. This data is summarized in Table 1.

Table 1. The standard deviation and mean of time between update packet in (μs)

	STD(μs)	Mean(μs)
Protocol 2	189.08	5000253.07
Protocol 3	420.14	5000279.08
Protocol 4	229.50	5000997.98

All three protocols had a standard deviation significantly lower than the mean. This means that all protocols considered would allow and ICD to maintain compliance with Galitz's rule. Early status updates can compromise this compliance but can easily be handled using a single stage queue to buffer date allowing the UI to update at its own defined interval. Thus, the user shouldn't not be annoyed by variable display update rates causing them to ignore the UI element representing the status data.

7. FURTHER WORK

This work has tested several protocols for application in wireless home networks but has not considered the all relevant effects of wireless networks, notably high packet loss rates, these should be considered in future work. While the robustness considerations presented will not impact the basic performance of these protocols, they should be implemented and thoroughly tested. Furthermore, these protocols, though targeted at UI updates should also be well suited to M2M communications between IoT device and this will be further considered.

The proposed protocols have been tested and assessed for application in a wireless home network however they may equally be considered for other networks (e.g. mesh or LoRa) and other applications and environments like enterprise, industrial or medical. While Protocol 3 is clearly best for wireless home environments other unique conditions may benefit other protocols and future work could consider these alternatives.

Security is an issue of significant concern for IoT networks, this work has maintained a solution that is as secure as the domestic Wi-Fi network used, however further investigation is warranted and would make for a valuable contribution in future work. Such work might apply a Red-Team style analysis where others are tasked with attempting to identify and exploit potential security vulnerabilities.

The work in this paper will inform the development of the other protocols required to support the elimination of the Permanent Central Controller from home automation systems. The next task to be considered is device display and control.

8. CONCLUSION

Home automation continues to be needlessly limited by the paradigmatic devotion to a Permanent Central Controller (PCC). Removing the PCC requires that it's functionality be distributed between IoT devices and Intermittent Controller Devices (ICDs) such as smartphones. This paper has considered one of those problems, how multiple ICD's can monitor devices in a packet efficient manner without a PCC. Four protocols were developed based on the ideas behind MQTT and CoAP. Assessment of the four proposed update protocols has shown all four to be preferable to UPnP and protocol 3 to be superior and worthy of full implementation. Further analysis has demonstrated how key additions can ensure the robustness and usability of any implementation of these protocols. It has therefore been demonstrated that without a PCC, multiple ICD's can maintain an up-to-date status of IoT devices without posing a significant load to the network or frustrating users.

This work lays the foundation for further development of IoT device Display and Control on ICD's which is the last set of protocols necessary to eliminate the PCC.

9. REFERENCES

[1] T. N. E. Steane and P. J. Radcliffe, "A universal iot joining protocol for DIY applications," *Proceedings of the 2017 27th International Telecommunication Networks and Applications Conference* , 2017, pp. 1–3.

[2] T. N. E. Steane and P. J. Radcliffe, "A novel discovery protocol for IoT based home automation," *International Journal of Automation and Smart Technology*, vol. 9, no. 3, pp. 147-158, 2019.

[3] T. N. E. Steane and P. J. Radcliffe, "Multiple intermittent controllers for IoT home automation," *Proceedings of the 2018 28th International Telecommunication Networks and Applications Conference*, 2018, pp. 1–6.

[4] "MQTT" [Online]. Available at: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[5] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)." [Online]. Available: https://tools.ietf.org/html/rfc7252.

[6] A. Panwar, A. Singh, R. Kumawat, S. Jaidka, and K. Garg, "Eyrie smart home automation using Internet of Things," *Proceedings of the 2017 Computing Conference*, 2017, pp. 1368–1370.

[7] P. Jutadhamakorn, T. Pillavas, V. Visoottiviseth, R. Takano, J. Haga, and D. Kobayashi, "A scalable and low-cost MQTT broker clustering system," *Proceedings of the 2017 2nd International Conference on Information Technology (INCIT)*, 2017, pp. 1–5.

[8] F. J. Bellido-Outeirino, J. M. Flores-Arias, E. J. Palacios-Garcia, V. Pallares-Lopez, and D. Matabuena-Gomez-Limon, "M2M home data interoperable management system based on MQTT," *Proceedings of the 2017 IEEE 7th International Conference on Consumer Electronics ICCE*, Berlin, 2017, pp. 200–202.

[9] R. K. Kodali and S. Soratkal, "MQTT based home automation system using ESP8266," *Proceedings of the 2016 IEEE Region 10 Humanitarian Technology Conference*, 2016, pp. 1–5.

[10] J. Prabaharan, A. Swamy, A. Sharma, K. N. Bharath, P. R. Mundra, and K. J. Mohammed, "Wireless home automation and security system using MQTT protocol," *Proceedings of the 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 2017, pp. 2043–2045.

[11] S. Spinsante *et al.*, "A LoRa enabled building automation architecture based on MQTT," *Proceedings of the 2017 AEIT International Annual Conference*, 2017, pp. 1–5.

[12] I. Florea, R. Rughinis, L. Ruse, and D. Dragomir, "Survey of standardized protocols for the Internet of Things," *Proceedings of the*

2017 21st International Conference on Control Systems and Computer Science (CSCS), 2017, pp. 190–196.

[13] L. Mainetti, V. Mighali, and L. Patrono, "An android multi-protocol application for heterogeneous building automation systems," *Proceedings of the 2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2014, pp. 121–127.

[14] O. Bergmann, K. T. Hillmann, and S. Gerdes, "A CoAP-gateway for smart homes," *Proceedings of the 2012 International Conference on Computing, Networking and Communications (ICNC)*, 2012, pp. 446–450.

[15] O. Bergmann, S. Gerdes, S. Schäfer, F. Junge, and C. Bormann, "Secure bootstrapping of nodes in a CoAP network," *Proceedings of the 2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2012, pp. 220–225.

[16] M. N. Sahana, S. Anjana, S. Ankith, K. Natarajan, K. R. Shobha, and A. Paventhan, "Home energy management leveraging open IoT protocol stack," *Proceedings of the 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, 2015, pp. 370–375.

[17] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Information centric networking in IoT scenarios: The case of a smart home," *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 648–653.

[18] S. Son, N. Kim, B. Lee, C. H. Cho, and J. W. Chong, "A time synchronization technique for coap-based home automation systems," *IEEE Transactions on Consumer Electronics*, vol. 62, no. 1, pp. 10–16, 2016.

[19] K. Hartke, "Observing resources in the constrained application protocol (CoAP)." [Online]. Available at: https://tools.ietf.org/html/-rfc7641#section-5.

[20] J. Joshi *et al.*, "Performance enhancement and IoT based monitoring for smart home," *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 468–473.

[21] UPnP Forum, "UPnP Device Architecture 2.0," 2015. [Online]. Available at: http://upnp.org/specs/arch/UPnP-arch-Device Architecture-v2.0.pdf.

[22] O. Tarasyuk, A. Gorbenko, V. Kharchenko, and T. Hollstein, "Contention window adaptation to ensure airtime consumption fairness in multirate Wi-Fi networks," *Proceedings of the 10th International Conference on Digital Technologies 2014*, 2014, pp. 344–349.

[23] A. Gorbenko, A. Romanovsky, V. Kharchenko, and O. Tarasyuk, "Dependability of service-oriented computing: Time-probabilistic failure modelling," *Proceedings of the Software Engineering for Resilient Systems*, 2012, pp. 121–133.

[24] R. Shingledecker, "Tiny Core Linux" [Online]. Available at: https://distro.ibiblio.org/tinycorelinux/welcome.html.

[25] https://bitbucket.org/tylersteane/dhap-iot-status-communication-for-home-automation

[26] The software that empowers, GNS3, [Online]. Available at: https://www.gns3.com/.

[27] W. O. Galitz, *The Essential Guide to User Interface Design: An Introduction to Guidesign Principles and Techniques*, Hoboken, John Wiley & Sons, Incorporated, 2007.

***Tyler Nicholas Edward Steane**, completed a BSc in Applied Physics and a BEng (Honours) in Electronic and Communications Engineering in 2015 at RMIT University in Melbourne.*

In 2016, he began work on a MEng in electrical and electronic Engineering and transferred to the PhD program in mid-2017. His research is focused on home automation and how ubiquitous smart homes might be realized.

***PJ Radcliffe**, received his B.Eng. from Melbourne University and his PhD from RMIT University, both in Australia.*

He is a senior lecturer in the School of Engineering at RMIT University.

His current interests include data driven education research, the Internet of Things, real time embedded systems, and data networks.