Advances in Internet of Things, 2015, 5, 27-38
Published Online October 2015 in SciRes. http://www.scirp.org/journal/ait
http://dx.doi.org/10.4236/ait.2015.54005

Home Automation Device Protocol (HADP): A Protocol Standard for Unified Device Interactions

Thomas Gonnot, Won-Jae Yi, Ehsan Monsef, Jafar Saniie

Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA
Email: tgonnot@iit.edu, wyi3@iit.edu, emonsef@iit.edu, saniie@iit.edu

Received 1 October 2015; accepted 27 October 2015; published 30 October 2015

Abstract

Daily activities have become more efficient and convenient with home automation. There are many different home automation devices in the market for consumers to enjoy their home for being smarter, resourceful, and remotely accessible. However, current home automation protocols lack extensibility and compatibility. In this paper, we propose a protocol standard for home automation system called Home Automation Device Protocol (HADP). This protocol standard aims for the interoperability of home automation devices across different platforms. Based on the IFTTT (IF-This-Then-That) model, we define a set of device communication protocols where devices' *triggers* and *actions* are combined to generate and manage interactions through a central node. The proposed protocol standard offers low power consumption and low bandwidth requirements using the minimum data packets to trigger an action on a home automation device. The protocol supports various communication mediums such as Wi-Fi, Bluetooth 4.2, ZigBee IP, 6LoWPAN, IEEE 802.15.4 standards, and Ethernet orany network layer supporting IPv6 protocol.

Keywords

Home Automation, IFTTT, Wireless Communication Protocol, Smart Living

1. Introduction

Internet of Things (IoT) is changing the way to interact and access the devices in various automated environment. The main objective of IoT is to enhance both device-to-device interactions, as well as device-to-human interactions via Internet [1]-[4]. A device is defined as any type of system which is able to send/receive data to other devices using various communication protocols [5]-[7]. Such devices can be a smartphone, light switch,

How to cite this paper: Gonnot, T., Yi, W.-J., Monsef, E. and Saniie, J. (2015) Home Automation Device Protocol (HADP): A Protocol Standard for Unified Device Interactions. *Advances in Internet of Things*, **5**, 27-38.
http://dx.doi.org/10.4236/ait.2015.54005

plant watering system, motion detection sensors, security system, or air conditioning system. IoT devices bring the convenience of controlling and monitoring devices from anywhere within the Internet connection. A highly promising application of IoT is in home automation [8]. Home automation enables devices inside the house to interact. By connecting and integrating various sensors and actuators, numerous applications can be implemented in home automation system. For example, a remote-enabled air conditioning system allows the room temperature to be adjusted based on users' presence [9]. Furthermore, being able to remotely control the lighting or security system can be useful to protect against burglary activities [10]. For home security, basic motion monitoring system with an alarm can be implemented to prevent any damage or forced-entry into the house.

There exists several home automation systems and each of them can utilize separate protocols such as Wi-Fi, ZigBee, Z-Wave, X10, Universal Powerline Bus (UPB) and more [11]-[16]. However, there is no common protocol standard for home automation to integrate devices from different manufacturers. A universal protocol standard for home automation system is required to enhance interoperability, flexibility and scalability of the system. Different types of wired/wireless communication protocols can be used for home automation including Wi-Fi, Bluetooth 4.2, ZigBee IP, 6LoWPAN, IEEE 802.15.4 standards and Ethernet [17]-[21].

The choice of communication medium depends on the type of sensors or actuators in home automation system. Nevertheless, the communication protocol must allow low energy consumption and sufficient data security. For wireless devices, low energy consumption is crucial for battery life. In particular for IoT devices, power consumption is even more critical. There are varieties of wireless protocols aimed for low power consumptions. ZigBee IP and Bluetooth 4.2 are available options that can be used for sensor nodes. For providing a low power direct Internet accessibility to the sensors, a protocol called 6LoWPAN can be considered. It implements the reduced version of IPv6 protocol for maximizing IEEE 802.15.4 standard's power and communication efficiency [22] [23]. The data communication medium is not limited to the wireless protocol. Wired connection like Ethernet can be also applied to the system. Furthermore, power lines can be used for communicating with other devices inside home [24].

Beside the communication protocol, home automation requires an approach to program interactions among different devices. In this paper, we utilize the notion of IFTTT (IF-That-Then-This) to address this issue. IFTTT is a web-based service that allows Internet users to create a chain-reaction from one web service application to another [25]. Based on a user-defined conditional statement, called a *recipe*, the *trigger* of one web service application activates an *action* of another web service application. For example, "if a user uploads a photo to cloud, then send a notification e-mail to family" is a possible *recipe* for IFTTT. The *trigger* in this example is "a user uploads a photo to cloud", and the *action* is "send a notification e-mail to family". The IFTTT terminology is described as shown in **Figure 1**.

The IFTTT model can be applied to home automation devices where one device can *trigger* the *action* of another device. **Figure 2** describes how home automation devices would react on the user-defined *recipes*. Two *recipes* are shown in **Figure 2**. First *recipe* is "If motion is detected in a room, then turn on the lights". When the motion sensor in the room detects a movement, it sends a *trigger* to the central node. Based on the *recipe* and the *trigger*, the central node sends an *action* to the room lights to turn on. Second *recipe* is "If temperature and humidity changes in the garden, then turn on the irrigation system". When the temperature and humidity sensor senses change, it sends the *trigger* to the central node. Then, the *trigger* is interpreted by the central node that sends an *action* to the irrigation system. These *recipes* can be generated by remotely accessing the central node of the home automation system, or it can also be accessed within the home network. The central node acts as a router for the home devices to access the Internet and integrates all different types of data communication mediums. Therefore, the central node offers a web interface to allow users to configure the different *recipes*, which can be accessed from computers, smartphones or tablets.

In this paper, we present a protocol standard for unifying home device interactions called Home Automation Device Protocol (HADP). This protocol is focused on enabling compatible connectivity of any home automation devices developed by different manufacturers. The protocol is based on the IFTTT (IF-This-Then-That) model, optimized for low power, and utilizes IPv6 UDP network protocol. The proposed home automation protocol provides the unified standard and techniques for connecting home automation devices, and user-defined home device interactions through the central node. This allows communication mediums such as Wi-Fi, Bluetooth 4.2, ZigBee IP, 6LoWPAN or Ethernet to be used for HADP.

This paper is organized as follows: Section 2 presents the full description of HADP including details of the protocol mechanism; Section 3 discusses the technical approach towards the reliability of HADP; Section 4 de-

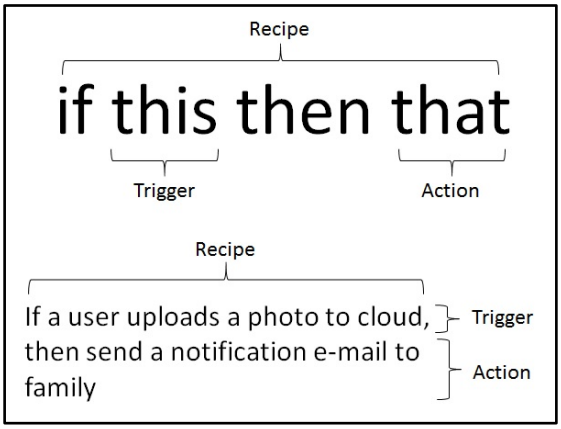


Figure 1. IFTTT description and example.

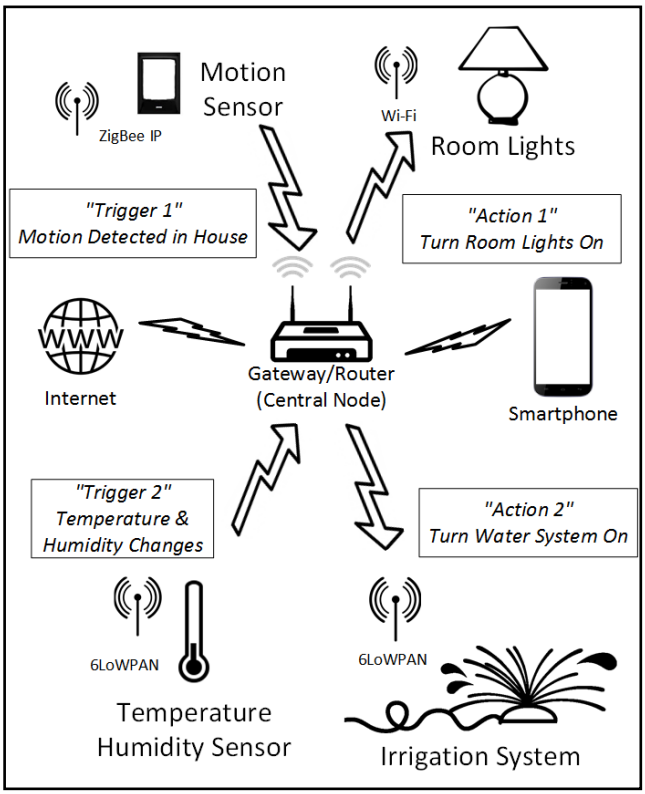


Figure 2. Home automation overview.

scribes the importance of Quality-of-Service (QoS) in HADP; Section 5 explains multi-hop routing scheme in HADP; Section 6 summarizes this paper.

2. Home Automation Protocol

The proposed HADP is designed to support the needs of the home automation network. It defines a set of rules which allows users to implement configurations over a set of interactions known as *recipes*. These *recipes* consist of associating *actions*/*triggers* to various devices in the home automation network. Therefore, each device requires to define and provide a set of *triggers* and a set of *actions* to be performed. Below, we present a complete description of the protocol, and define *triggers* and *actions* in detail. The proposed protocol is inspired by the CoAP protocol, designed for resource-constrained IoT devices [26]. However, CoAP is an evolution of the

HTTP protocol, and still requires the use of ASCII command to function. In contrast, HADP uses a specific packet format to operate, while improving the resource discovery on the home automation network.

2.1. Network Discovery

A new device upon connection to a network must register to the central node. Therefore, the new device sends a broadcast message called *Discovery* packet in order to find the central node which is followed by a reply from the central node. In this initial stage, the central node sends request to the device asking its features, which is a description of the different *triggers* and *actions* described below. The device formats this data in a *Descriptor* packet. Even though this packet can be potentially large, this will not impact on the device power consumption since it should only be sent once when initially introduced in the network.

2.2. Triggers

The *triggers* are sent from a device to the central node. It can be either an empty packet with a trigger flag and the trigger ID, or it can contain a certain payload, with a type defined in the device descriptor. In the case of a packet with embedded data, the central node can propose different actions to the user depending on the nature of the data and its value. For example, the *trigger* of a temperature sensor would have the measured temperature as part of the packet's data.

2.3. Actions

The *actions* are controlled from the central node. As for the *trigger*, the packet can be empty with just an action flag and the action ID, or contain a payload. The payload can be used to configure the *action* to defined parameters. For example, the *Action* packet for a RGB light bulb would contain the new color intensities for red, green and blue.

2.4. Packet Format

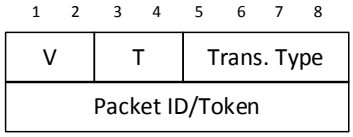
All packets start with two bits defining the version of the protocol (00 for this revision), and two bits defining the packet type. The protocol is composed of four different types of packets: *Connection*, *Descriptor*, *Trigger* and *Action*.

2.4.1. Connection Packet

The *Connection* packet is used to handle the connection between the central node and the home device. It is used by devices to discover the available central node(s), perform connection request and confirmation, and finally generate acknowledgement packets. **Figure 3** shows the organization of the *Connection* packet. The "V" is a two-bit field representing the version of the protocol. The "T" is also a two-bit field representing the packet type. Both fields are set by default to 00 for a *Connection* packet. The rest of the first byte is composed of four bits defining the transaction packet type:

- The *Acknowledgment* packet is used to acknowledge the reception of a packet that requires acknowledgement. The next byte in the header is the packet ID (3 bits).
- The *Discovery* packet is used to send a discovery request on the network. This packet should be sent by the device in multicast over the network. Next byte is a token for the request.
- The *Offer* packet is used by the central node to respond to a *Discovery* request and offers the device to connect. The next byte is the token from the *Discovery* request.
- The *Confirm* packet is used by the device to confirm the connection to a central node. The token found in the *Discovery* and *Offer* packets is included in the data, as well as the Vendor ID, Product ID and Serial number of the device that is unique as shown in **Figure 4**.

The connection process is shown in **Figure 5**. Two cases are represented in this figure, one including the request for a *Descriptor* by the central node in the acknowledgement packet, and the other without request for a *Descriptor*. The *Descriptor*, defined in Section 2.4.2., can contain a large quantity of data, and therefore the central node should request only once. Devices are assigned with Vendor ID, Device ID and Serial number so that the central node can determine whether or not the device was previously connected. If so, the *Descriptor* is not requested.



Transaction Type:
- 0000: Reserved
- 0001: Discover
- 0010: Offer
- 0011: Confirm
- 0100: ACK
- 0101: ACK + Request
- 0110: QoS Update
- 0111: Config Update

Figure 3. *Connection* packet.

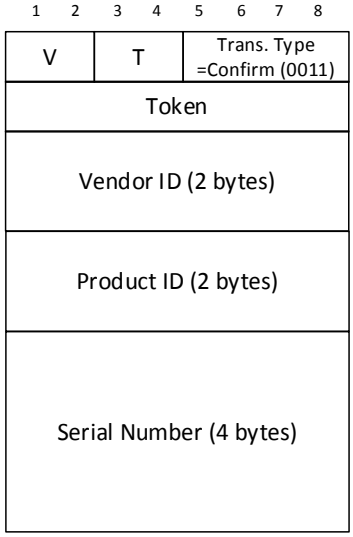


Figure 4. Additional data for the *confirm* packet.

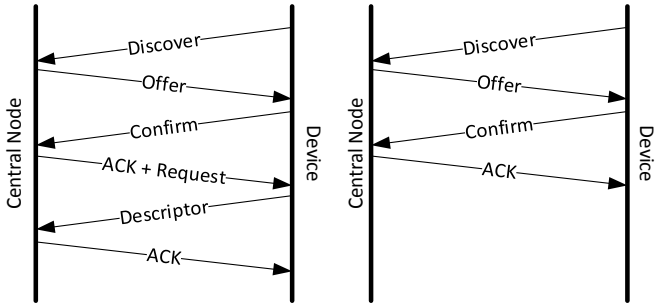


Figure 5. Connection sequence.

In order to support basic Quality-of-Service (QoS), the *Connection* packet supports the transmission of QoS related data, with the packet format defined in **Figure 6**. Several flags indicate the content of the packet:

- BW: Indicates that a bandwidth/throughput requirement is embedded in the data;
- BER: Indicates that a Bit Error Rate information is embedded in the data;
- Prio.: Indicates that a priority information is embedded in the data;
- Delay: Indicates that a maximum delay requirement is embedded in the data;
- Batt.: Indicates that a battery level information is embedded in the data.

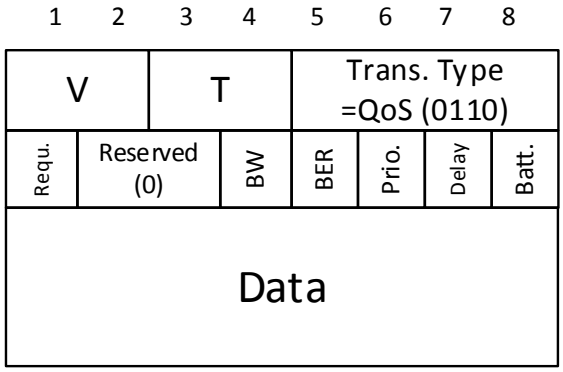


Figure 6. Quality-of-Service packet.

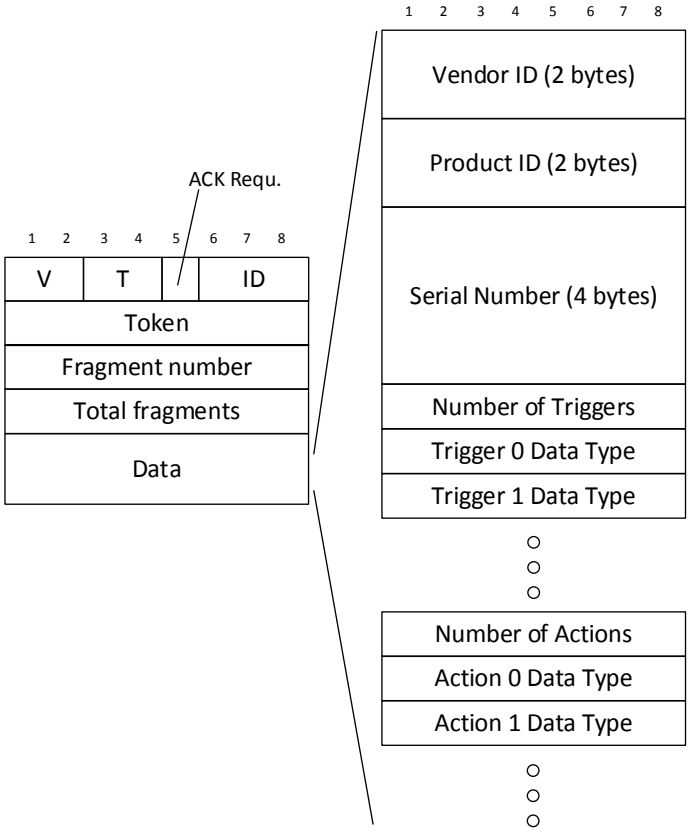


Figure 7. *Descriptor* packet.

An extra Request flag is available so that the central node can force the nodes to transmit the data. Data selection is done by enabling each individual flag of the required data. No acknowledgement is required for this packet.

2.4.2. Descriptor Packet

The *Descriptor* packet is used to transmit the different features of the device. It contains all the information about the different *triggers* and *actions*, and the type of data they support. **Figure 7** shows the organization of the *Descriptor* packet. The version V remains the same value as before (00) and the type T is set to 01. The header also includes a flag for acknowledgement request, and a packet ID over 3 bits. A token is used to identify the *Descriptor* across different packets. The next two bytes indicates the packet number and the total number of packets for the Descriptor.

The transmitted data contains the Vendor ID, Product ID and Serial number of the device. Then, it is followed by the number of *triggers*, and the description of each *trigger*, and then the number of *actions* and the description of each *action*.

2.4.3. Data Types

The data type includes information about the data transmitted, with not only the type of data, but also can include a small description, formatted as shown in **Figure 8**. This helps the user to understand what each *trigger* and *action* is. Each type description is determined by one byte defining the data type (see **Table 1**), followed by a short description encoded in UTF-8.

A flag allows to define the data as being an array of variables of the same type. In this case, the data is preceded by one byte defining the number of elements on the array in the following packets for *trigger* and *action*. The UTF-8 string is a natural array and its format is also preceded by the size of the string, but over 2 bytes.

2.4.4. Trigger Packet

The *Trigger* packet is sent by the device to the central node to report an event or a value. The packet is organized as shown in **Figure 9**. The version V is set to 00, and the type T is set to 10, followed by a flag for acknowledgement request, and a packet ID over 3 bits.

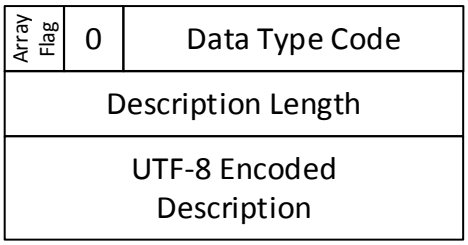


Figure 8. Data type and description format.

Table 1. Different data types and corresponding codes.

Code	Data Type
000000	Boolean
000001	Char
000010	Unsigned char
000011	Short
000100	Unsigned short
000101	Int
000110	Unsigned int
000111	Long
001000	Unsigned long
001001	128 bits
001010	Unsigned 128 bits
001011	256 bits
001100	Unsigned 256 bits
001101	Float
001110	Double
001111	UTF-8
010000	UTF-8 String
010001	Localization (Country/Region)
010010	Localization (GPS)
010011	Time
010100	Forecast Summary

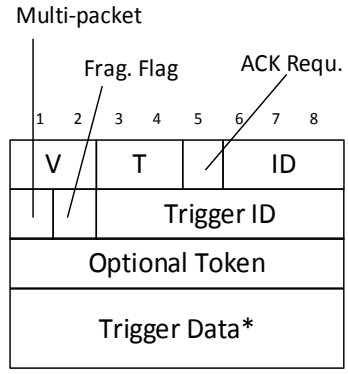


Figure 9. *Trigger* packet.

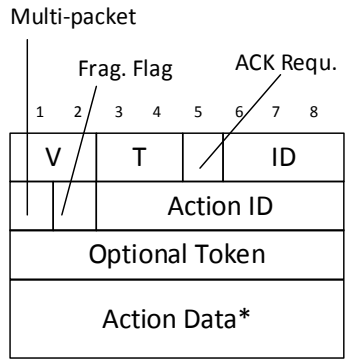


Figure 10. *Action* packet.

The second byte includes two flags:

- The Multi-packet flag indicates that the packet includes data for different devices, and that data needs to be relayed. This functionality is described in Section 2.4.6;
- The Frag. flag indicates that the packet is divided across several packets. This can replace or complement the eventual lower layers of the network stack. When this functionality is used, a token is inserted to keep track of the data between the different packets.

The remaining 6 bits are used to indicate the Trigger ID, which defines the type of data that is transmitted, if any. The type of data transmitted needs to match the one described in the *Descriptor*.

2.4.5. Action Packet

The *Action* packet is sent by the central node to device. The packet is virtually identical to the *Trigger* packet, as shown in **Figure 10**, with the difference of having the type T set to 11.

The second byte includes two flags:

- The Multi-packet flag indicates that the packet includes data from different devices, and that data needs to be relayed to the central node. This functionality is described in Section 2.4.6;
- The Frag. flag indicates that the packet is divided across several packets. It has the same purpose than for the Trigger packets.

The remaining 6 bits are used to indicate the Action ID, which defines the type of data that is transmitted, if any. The type of data transmitted needs to match the one described in the *Descriptor*.

2.4.6. Packet Concatenation

Several *actions* and *triggers* can be concatenated in a single packet in an attempt to save power by reducing the number of transmitted packets. The functionality differs between the *actions* and the *triggers*. The central node can decide to pack a certain number of *actions* in a single packet, to be sent to several nearby devices on the network. This allows to minimize the number of packets transmitted through the network.

The *actions* are read sequentially and therefore the efficiency of this method depends on the capability of the

central node to determine an optimum route. In this case, the central node also expects only one acknowledgement from the last node addressed in the packet, if requested.

The *triggers* concatenation is managed by the devices relaying the data to the central node. The behavior is more opportunistic, and consists of adding its own *trigger* to the packet if the device was about to send it on another packet. In this case, the devices still require separated acknowledgement packets, but a device adding data to the packet can eventually acknowledge the data in place of the central node, and take care of the retransmission in case the central node doesn't acknowledge the data.

When the Multi-packet flag is set, the data is preceded by the offset for the next data in bytes. Each data is also followed by the IP of the next device. The IPv6 device addresses are compressed using the method defined by the 6LoWPAN protocol. **Figure 11** shows a representation of the data organization for a concatenated packet.

3. Reliability in HADP

Reliability of data transmission is crucial in home automation application. To date, several techniques have been designed to address the reliability of packet transmission in TCP/UDP protocols. However, these protocols are not lightweight and therefore not suitable for home automation applications. The reliable transmission of a packet in the proposed protocol is initiated by marking the packet as "acknowledgeable" in the HADP header. An acknowledgeable packet always specifies that it requires confirmation from the receiver by setting the ACK flag. The receiver is required to confirm the reception of the packet by sending an acknowledgement packet. The acknowledgement packet must echo the packet ID of the acknowledgeable packet. In case the sender does not receive any reply from the receiver, it retransmits the acknowledgeable packet until it receives a confirmation packet or runs out of attempts. Retransmission is controlled by two parameters: a timeout and a retransmission counter. HADP node needs to keep track of these values for each sent packet while waiting for an acknowledgement. For a new acknowledgeable packet, the timeout is set to a random value called TIMEOUT_RETRANSMIT and the retransmission counter is set to the value zero. When the timeout is triggered and the retransmission counter is less than TIMEOUT_RETRANSMIT, the packet will be retransmitted with the same ID and the retransmission counter is incremented. If the retransmission counter reaches TIMEOUT_RETRANSMIT on a timeout for a maximum number of retransmission, then the sender will stop sending the packet and inform the process of failure in delivery. Otherwise, if the endpoint receives an acknowledgement in time, the packet transmission is considered successful. This approach ensures the reliability of message and can be implemented on resource-constraint devices.

4. Quality of Service in HADP

Providing Quality-of-Service (QoS) is among the most important requirements of multimedia services for the home automation networks. The increase of demands for high bandwidth devices such as high quality Audio/Video, streaming, Voice over IP (VoIP) and Internet TV, requires fair allocation of network resources among

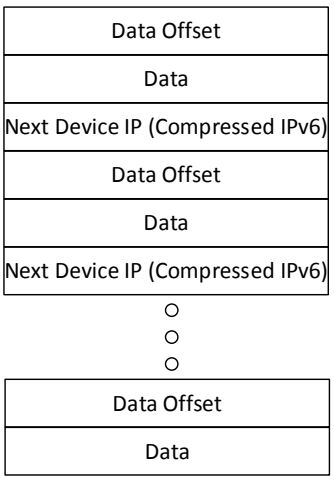


Figure 11. Data organization for concatenated packet.

different devices inside homes. Furthermore, the delay sensitive devices such as VoIP phones make the QoS implementation in home automation more appealing. However, current QoS frameworks, such as Universal Plug and Play QoS architecture [27], do not provide an online and real-time integrated QoS mechanism for the high bandwidth and low latency devices. In a successful QoS design, a central scheduler is required to allocate resources given the specific requirement of the devices in a given time. One merit of the proposed HADP protocol is using a dedicated QoS packet in order to report the resource requirement of a device to the central scheduler. Therefore, it is possible to utilize this packet to report the QoS requirement as well. Whenever a device needs to update its QoS requirements, it will immediately send a QoS packet. Thus, at each epoch of time, the resource scheduling inside the central node will allocate the resources according to current real-time QoS requirements. The content of a QoS packet will be customized based on the nature of the home automation application using a set of flags defined previously (see protocol descriptions in Section 2). Four different QoS parameters are described below, which will be reported to the central node by each sensor node.

- Delay: The packet delivery time is crucial mainly for multimedia applications in home automation where only few milliseconds of delay could result in service disruption. The delay often due to the queuing of the data or the transmission delay.
- Bit Error Rate (Wireless Quality): Reliability of data transmission could be affected by the ratio of the number of error bits received at the receiver to the number of bits generated by the sender known as Bit Error Rate (BER). Higher BER could result in the waste of valuable energy since the packet needs to be re-transmitted.
- Energy Consumption: It is important to allocate resource with the goal to minimize the overall energy consumption of the devices. Besides, some devices are battery-powered and their energy needs to be saved for long time.
- Throughput: Devices require homogeneous packet delivery ratio in a home automation which varies from few bits per second to several megabits per second depending on the application requirements.

5. Multi-Hop Routing

The application of home automation spans from a small-size house to large commercial buildings. Since the HADP design is a centralized solution, it mainly relies on the central node to perform resource allocation and scheduling. Therefore, providing signal coverage for large-scale scenarios becomes a challenging task and requires careful planning. Furthermore, a pure centralized approach may increase the overall energy consumption of the system as the energy depletion due to transmission increases exponentially with distance between the sender and the receiver. In order to overcome this challenge, we propose to utilize the wireless multi-hop routing between devices. This approach is similar to IPv6 header characteristic where the options are linked together in the format of a linked list. Suppose a central node needs to send a message to node A and B which are in the vicinity of each other while the device A is closer to the central node. Then, central node will create a concatenated packet as shown in **Figure 12**.

Whenever the device A receives a multi-hop packet, it pops the top part of the message body and sends the remaining part of the message to the next device. Two issues may arise in this setting. First, the existence of multiple messages inside a single packet will increase the size of the packet and therefore waste the energy consumption of intermediate nodes. Furthermore, security is a big challenge in this approach as devices are able to observe or modify the messages belonging to other devices. In order to tackle the first issue we propose to use the IPv6 address compression which has been implemented in 6LoWPAN [28]. Moreover, in the case of 6LoWPAN, this mechanism is implemented using the Full-Function Devices (FFD), whereas the devices with power

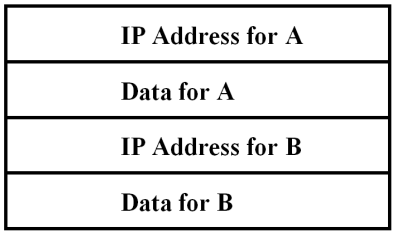


Figure 12. Packet concatenation for multi-hop routing.

of processing constrains are left as Reduced-Function Devices (RFD) and kept out of the multi-hop routing. As for the security issue, the suggested solution is to encrypt or sign the data contained in the message when necessary.

6. Conclusion

In this paper, we proposed the Home Automation Device Protocol (HADP) to allow scalability, extensibility and compatibility of home automation systems. Using the proposed home automation protocol standard, we incline to transcend the barrier of incompatibility across the home automation platforms. This paper proposes a device communication protocol based on the IFTTT model with *triggers* and *actions* defined for each device and managed using a central node. The proposed protocol standard operates with four different types of packets designed to allow low power consumption and low bandwidth requirements on the wired/wireless mediums. The use of the IPv6 allows any supported medium to work with HADP. Examples of HADP-enabled mediums include Wi-Fi, Bluetooth 4.2, ZigBee IP, 6LoWPAN and IEEE 802.15.4 standards.

References

[1] Tech Target (2014) What is Internet of Things (IoT)?—Definition from WhatIs.com. http://whatis.techtarget.com/definition/Internet-of-Things

[2] O'Reilly Radar (2014) #IoTH: The Internet of Things and Humans. http://radar.oreilly.com/2014/04/ioth-the-internet-of-things-and-humans.html

[3] Studio Science (2015) Interaction Design within the Internet of Things. http://studioscience.com/interaction-design-within-the-internet-of-things/

[4] Siorpaes, S., Broll, G., Paolucci, M., Rukzio, E., Hamard, J., Wagner, M. and Schmidt, A. (2006) Mobile Interaction with the Internet of Things. *Adjunct Proceeding of Pervasive* 2006 *Late Breaking Results*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.449.4139&rep1&type=pdf

[5] Das, S.R., Chita, S., Peterson, N., Shirazi, B. and Bhadkamkar, M. (2011) Home Automation and Security for Mobile Devices. 2011 *IEEE International Conference on Pervasive Computing and Communications Workshops* (*PERCOM Workshops*), Seattle, 21-25 March 2011, 141-146. http://dx.doi.org/10.1109/PERCOMW.2011.5766856

[6] Al-Ali, A.R., Qasaimeh, M., Al-Mardini, M., Radder, S. and Zualkernan, I.A. (2015) ZigBee-Based Irrigation System for Home Gardens. 2015 *International Conference on Communications, Signal Processing, and Their Applications* (*ICCSPA*), Sharjah, 17-19 February 2015, 1-5. http://dx.doi.org/10.1109/ICCSPA.2015.7081305

[7] Safe Sound Family (2015) Best Smart Home Security Systems: DIY Home Automation Security Reviews, Recommendations. http://safesoundfamily.com/blog/50-best-smart-home-security-products/

[8] Read Write (2013) How Big The Internet of Things Could Become. http://readwrite.com/2013/09/30/how-big-the-internet-of-things-could-become

[9] Nest (2015) Works with Nest. https://nest.com/works-with-nest/

[10] Maiti, A. and Sivanesan, S. (2012) Cloud Controlled Intrusion Detection and Burglary Prevention Stratagems in Home Automation. 2012 2*nd Baltic Congress on Future Internet Communications (BCFIC)*, Vilnius, 25-27 April 2012, 182-186. http://dx.doi.org/10.1109/BCFIC.2012.6218000

[11] Digital Trends (2014) Zigbee vs. Z-Wave vs. Insteon: Home Automation Protocols Explained. http://www.digitaltrends.com/home/zigbee-vs-zwave-vs-insteon-home-automation-protocols-explained/

[12] Engadget (2014) Nest, Samsung and Others Team up for Better Home Automation. http://www.engadget.com/2014/07/15/google-samsung-arm-thread-connected-home-iot-standard/

[13] SecurityGem (2015) Does This Home Automation Product Work with That? http://securitygem.com/home-automation-product-work/

[14] Belkin (2015) WeMo Home Automation. http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/

[15] Twice (2015) Comcast Home Automation System Opens up. http://www.twice.com/news/home-automation/comcast-home-automation-system-opens/57077

[16] Simply Automated Incorporated (2015) Home Automation Made Simple. http://www.simply-automated.com/

[17] Piyare, R. and Tazil, M. (2011) Bluetooth Based Home Automation System Using Cell Phone. *Proceedings of the* 2011 *IEEE* 15*th International Symposium on Consumer Electronics* (*ISCE*), Singapore, 14-17 June 2011, 192-195. http://dx.doi.org/10.1109/ISCE.2011.5973811

[18] Dominguez, F., Touhafi, A., Tiete, J. and Steenhaut, K. (2012) Coexistence with WiFi for a Home Automation ZigBee Product. *Proceedings of the* 2012 *IEEE* 19*th Symposium on Communications and Vehicular Technology in the Benelux* (*SCVT*), Eindhoven, 16 November 2012, 1-6. http://dx.doi.org/10.1109/SCVT.2012.6399392

[19] Olteanu, A.-C., Oprina, G.-D., Tapus, N. and Zeisberg, S. (2013) Enabling Mobile Devices for Home Automation Using ZigBee. *Proceedings of the* 2013 *19th International Conference on Control Systems and Computer Science* (*CSCS*), Bucharest, 29-31 May 2013, 189-195. http://dx.doi.org/10.1109/CSCS.2013.63

[20] Radio-Electronics.com (2015) What Is Zigbee IP | IPv6 Mesh Networking | Tutorial. http://www.radio-electronics.com/info/wireless/zigbee/zigbee-ip.php

[21] Bluetooth SIG (2015) Bluetooth 4.2 Smarter, Faster, Enables IoT | Bluetooth Technology Website. http://www.bluetooth.com/SiteCollectionDocuments/4-2/bluetooth4-2.aspx

[22] Gonnot, T. and Saniie, J. (2014) User Defined Interactions between Devices on a 6LoWPAN Network for Home Automation. *Proceedings of the* 2014 *IEEE International Technology Management Conference* (*ITMC*), Chicago, 12-15 June 2014, 1-4. http://dx.doi.org/10.1109/ITMC2014.6918618

[23] Tudose, D.S., Voinescu, A., Petrareanu, M., Bucur, A., Loghin, D., Bostan, A. and Tapus, N. (2011) Home Automation Design Using 6LoWPAN Wireless Sensor Networks. *Proceedings of the* 2011 *International Conference on Distributed Computing in Sensor System and Workshops* (*DCOSS*), Barcelona, 27-29 June 2011, 1-6. http://dx.doi.org/10.1109/DCOSS.2011.5982181

[24] Dange, H.V. and Gondi, V.K. (2011) Powerline Communication Based Home Automation and Electricity Distribution System. *Proceedings of the* 2011 *International Conference on Process Automation*, *Control and Computer* (*PACC*), Coimbatore, 20-22 July 2011, 1-6. http://dx.doi.org/10.1109/PACC.2011.5978900

[25] Mashable (2013) How IFTTT Is Changing the Way We Do Things on the Web. http://mashable.com/2012/12/04/ifttt/

[26] Shelby, Z., Hartke, K. and Bormann, C. (2014) The Constrained Application Protocol (CoAP). Internet Engineering Task Force (IETF) RFC-7252. http://dx.doi.org/10.17487/RFC7252

[27] Zhang, Y., Yu, R., Xie, S., Yao, W., Xiao, Y. and Guizani, M. (2011) Home M2M Networks: Architectures, Standards, and QoS Improvement. *IEEE Communications Magazine*, **49**, 44-52. http://dx.doi.org/10.1109/MCOM.2011.5741145

[28] Kawamura, S. and Kawashima, M. (2010) A Recommendation for IPv6 Address Text Representation. IETF RFC 5952, August 2010.