ASUT : Advanced Software Update for Things

Juhyun Choi

Samsung Electronics and
Sungkyunkwan University
Suwon, South Korea
Email: honest.choi@samsung.com

Changue Jung, Ikjun Yeom and Younghoon Kim

Sungkyunkwan University
Suwon, South Korea
Email: {ckjung1987, ijyeom, kyhoon}@gmail.com

***Abstract*—Due to severe competition among manufacturers, timely firmware updates have become one of the most important issues. The constantly increasing number of things connected to the Web leads to high traffic contention in shared networks that causes poor service quality. In this paper, we propose an efficient software update system for smart things. The burden of downloading updates is offloaded to an always-on in-home hub. This delegation achieves not only avoidance of contention, but also the decrease of unnecessary transfer requests. Updates are transferred to the registered smart things without harming active service traffic. To achieve these goals, we implement a transport scheme based on Quick User Datagram Protocol (UDP) Internet Connections (QUIC) protocol that is known as emerging transport layer for Hypertext Transfer Protocol (HTTP). Our experimental results show that the proposed scheme is completely backed off with the existence of active service traffic and quickly completes the transfers when no others are active.**

Keywords–protocol; transport; software update.

I. INTRODUCTION

Recently, the rapid growth of smart things enforces manufacturers to be in a hurry when releasing their products. As a result, flawed software is often shipped and on-time software updates become one of the most urgent and important issues among manufacturers and service providers. Although both online and offline updates are possible, updates through the Internet occupy a dominant portion thanks to the development of network infrastructure and easier user scenarios for deploying them. A naive server-client communication model is commonly used for updating things, and updating scenarios can be categorized in two distinctive ones. In the first category, downloading firmware and/or applying it occurs when the things are in standby mode. Timely firmware updates for things, however, become hard to be achieved in this scenario due to efforts for reducing standby power based on this report [1]. So, not-in-use things would be unplugged and updating in the standby mode would not be realized. In the second scenario, things are updated only when they are in-use. Downloading and updating are initiated when users actually use the things. Regarding large sizes of firmware, however, it makes updating procedures unreliable due to unpredictable users' on-off patterns. Partial firmware updating is one alternative, but it is not considered as a realistic option because of its high implementation complexity. In case of always-on small things, keeping a connection for updating is a burden because of insufficient processing power. Also, in the view of service quality with shared network, numerous Hypertext Transfer Protocol (HTTP) requests congest the network. This contention causes a fluctuation in request latency.

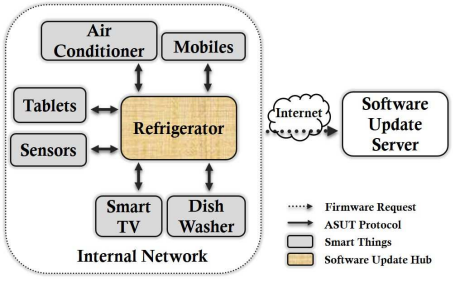


Figure 1. Overall ASUT Architecture

In this paper, we consider an alternate efficient firmware updating system for smart things. In our proposed scheme, an always-on thing (a refrigerator in this paper) is exploited as a software update hub onto which smart things can offload their firmware downloading. Once the hub completes downloading firmware, it starts updating things. As mentioned above, the size of firmware becomes large and transferring it with conventional transport protocols may cause severe contention between things' traffic and firmware transfer. Under the assumption that in-home network is separated from the Internet, a new transport protocol is proposed to mitigate this contention. The proposed protocol achieves high transfer rate in idle network and uses a quick backoff algorithm not to harm the users' quality of network usage. Another advantage of our proposed scheme is that stored firmware can be reused when identical things reside in shared network. This aspect not only enhances firmware accessibility but also lowers update servers' loads which suffer from repeated update requests.

The rest of this paper is structured as follows. In Section 2, we illustrate the design of our proposed scheme. Section 3 shows the result of the proof-of-concept tests. Finally, in Section 4, the conclusion of our work is described.

II. ASUT DESIGN

In perspective of network, a network software update can be treated as one large file transfer that needs high reliability. But, the conventional network update scheme has no way to consider the firmware transfer in a special manner. To transfer firmware efficiently with less damage to active services, we designed a system as described in Figure 1, and named it Advanced Software Update for Things (ASUT).

ASUT is an advanced network update system for smart things. ASUT offloads firmware downloading to the dedicated hub residing in home and efficiently distributes it to internal devices without degrading the network quality of running services. All firmware requests are delegated to the software

 ISBN: 978-1-61208-598-2

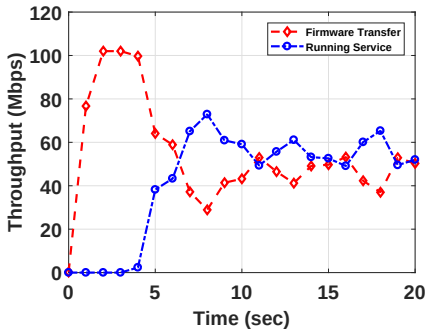


Figure 2. Contention Simulation in Conventional Network

update hub, which has to be an always-on thing, normally a refrigerator. To avoid congestion, firmware downloading is postponed until the network is idle. This centralization achieves not only high utilization of bandwidth, but the decrease of redundant transfers from identical things. For the distribution of downloaded firmware, we modified Quick User Datagram Protocol (UDP) Internet Connections (QUIC) [2] protocol. In order to steer the aggressiveness of the transfer, we utilize the combination of the number of the virtual connections and pacing mechanism [3]. The number of virtual connections implies the aggressiveness of a flow, and we manipulate it to control the aggressiveness in a coarse-grained manner. Setting a larger number of virtual connections empowers aggressiveness to the flow. So, our protocol dynamically adjusts it based on the number of packet loss.

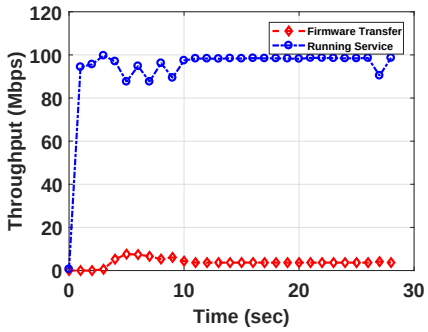
To manage aggressiveness in a fine-grained manner, pacing is employed. Pacing is aware of the time intervals between two consecutive packet-send-events, and it determines when to send a next packet. We exploit pacing to make ASUT to be conservative when there are other active flows in the same network. Otherwise, ASUT should utilize maximum bandwidth. Specifically, we let the pacing rate increase after a large number of successful packet transfers (slow increase) and we let it decrease sharply with only a few packet losses (fast backoff). These features are designed to achieve the maximum throughput in idle conditions, while staying at a minimum during contention.

III. Preliminary Evaluation

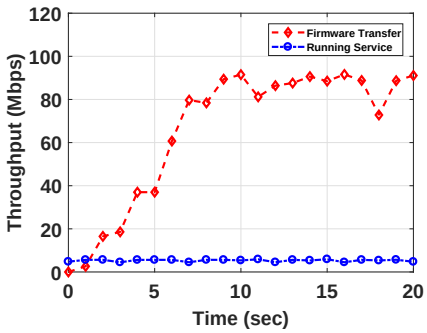
In this section, we will show the result of proof-of-concept tests to verify the ASUT transport algorithm. The test is conducted in a dumbbell topology that employs a bottleneck link featuring 100Mbps and a queue size of bandwidth delay product. Our server and client implementation are based on proto-quic [4], and Ubuntu 14.04 distribution is installed on nodes. One server mimics the software update hub in the test, and the other one acts as contending running services.

Figure 2 shows how the conventional firmware transfer competes with an existing service flow. As soon as the transfer starts, the existing flow immediately yields the bandwidth. This implies existing HTTP requests are delivered with high latency. And, if the active traffic is for video streaming, this bandwidth sharing may lower streaming bitrate or cause the distortion of video streaming, which severely spoils user experience.

The test in Figure 3 illustrates the effect of ASUT. Once the network is dominated by the running service, the transport protocol of ASUT operates to maintain the minimum bandwidth as shown in Figure 3(a). If packet losses are detected frequently, ASUT decreases the pacing rate that contributes to decrease



(a) Less Damages in Contention with ASUT



(b) Efficient Transfer in Idle Network with ASUT

Figure 3. Contention with ASUT

the bandwidth. On the other hand, the maximum bandwidth is achieved in idle network as shown in Figure 3(b). Once the flow maintains high pacing rate without the packet losses, ASUT accelerates the flow by the increase of the number of virtual connections.

IV. Conclusion

We proposed an efficient and reliable software update system for smart things, named ASUT. In ASUT, jobs for downloading software updates are offloaded to an in-home hub and those updates are transferred to things without harming other active HTTP traffic. A transport protocol, designed based on QUIC, achieves quick transfer between the hub and devices with no harm to other protocols or services. Our ASUT helps both network utilization and service quality in a flood of smart things. For the future works, we have plans to precisely design the communication protocol and implement ASUT on real devices.

Acknowledgment

This work was supported by National Research Foundation (NRF) of Korea grant funded by the Korea government (MSIP) (NRF-2016R1E1A1A01943474 and NRF-2016R1C1B1011682).

References

[1] Harrington et al., “Standby energy: Building a coherent international policy frameworkmoving to the next level,” Stockholm: European Council for an Energy Efficient Economy, 2007.

[2] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, “Quic: A udp-based secure and reliable transport for http/2,” IETF, draft-tsvwg-quic-protocol-02, 2016.

[3] A. Aggarwal, S. Savage, and T. Anderson, “Understanding the performance of tcp pacing,” in INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3. IEEE, 2000, pp. 1157–1165.

[4] [Online]. Available: https://github.com/google/proto-quic [retrieved: October, 2017]

 ISBN: 978-1-61208-598-2