International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Challenges in IoT Networking via TCP/IP Architecture

T Ramya[1], K Anitha[2]

[1, 2]Lecturer,Dept. of Computer Systems and Engineering, Loyola Academy Degree and PG College, Alwal, Secunderabad, 500010, Telengana, India

Abstract: ***"Internet of Things" (IoT), networking (potentially) a large number of resource-constrained devices, is gaining popularity in recent years. Today's IoT systems are largely based on the use of the TCP/IP protocols (IPv6 in particular). However, the observations so far suggest that the TCP/IP protocol stack, as originally designed, is not a good fit to the IoT environment. Over the last several years the IETF has spent significant amount of effort in modifying the protocol stack to fit IoT deployment scenarios. These efforts have resulted in extensions to existing protocols in the TCP/IP protocol suite as well as development of multiple new protocols. Yet new problems continuously occur. In this paper we analyze the technical challenges in applying TCP/IP to the IoT environment and review various solutions proposed by the IETF. We argue that existing IP-based solutions are either inefficient or insufficient in supporting IoT applications, and that a more effective solution would embrace the Information Centric Network architecture.***

Keywords: Internet of Things; TCP/IP; network architecture

1. Introduction

"Internet of Things" (IoT) generally refers to the interconnection of different types of computing devices to support various kinds of monitoring and control applications. To accommodate the heterogeneity of devices and applications from different venders, modern IoT systems have adopted the open standards of TCP/IP protocol suite, which was developed for the wired global Internet several decades ago, as the networking solution. However, IoT networks differ from traditional wired computer networks in fundamental ways as we elaborate below. Those differences pose significant challenges in applying TCP/IP technologies to the IoT environment, and addressing these challenges will make a far-reaching impact on the network architecture. This paper aims to systematically identify the challenges posed by the IoT environment, and to articulate the future direction to tackle the challenges. IoT networks often contain a large number of low-end, resource-constrained devices. The design of those devices are mostly driven by low manufacturing and operational cost. As a result, the IoT devices are typically equipped with limited computing power and required to operate over long time periods (e.g., a year) on battery. Due to the power constraints, the IoT networks often employ low-energy Layer-2 technologies, such as IEEE 802.15.4, Bluetooth LE and lowpower Wi-Fi, which usually operate with much smaller MTU and lower transmission rate compared to traditional Ethernet links. Therefore an immediate challenge for the IoT network protocol design is to adapt the packet size to the constrained links (discussed in Section 2.1). To save energy, IoT nodes may not be always on as in wired networks. Moreover, an IoT system may be deployed in environments without wired network infrastructure (e.g., forests, underwater, battle fields) and consequently has to rely on wireless mesh technologies to communicate. This brings more challenges to the TCP/IP protocol architecture: first, mesh networks typically adopt the multi-link subnet model which is not supported by the original IP addressing architecture (discussed in Section 2.2); second, broadcast and multicast are expensive on a battery powered network as a single multicast will involve a series of multi-hop forwarding and potentially wake up many sleeping nodes (discussed in Section 2.3); third, a scalable routing mechanism is now necessary for IP communications to happen over the mesh networks (discussed in Section 2.4); and lastly, the TCP-style reliable and in-order byte stream delivery is often ill-suited for applications that require customized control and prioritization of their data (discussed in Section 3). Most IoT applications interact with lots of sensors and actuators to perform various monitoring and control tasks on the ambient environment. Their design patterns intrinsically require efficient and scalable support for naming con- figuration and discovery, security protection on the data acquisition and actuation operations, and a resource-oriented communication interface such as Representational State Transfer (REST). Unfortunately, existing solutions to those problems, many of which are widely used by today's Web technologies, do not satisfy the constraints of the IoT environments. For example, the traditional DNS-based naming services are unsuitable in many IoT deployment scenarios that lack infrastructural support for dedicated servers (see Section 4.1). The application-layer content caches and proxies are often inefficient in dynamic network environments with intermittent connectivity (discussed in Section 4.2). In addition, the channel-based security protocols such as TLS and DTLS, which are used to secure the REST communications, impose high overhead on the IoT devices in terms of protocol operations and resource consumption (discussed in Section 4.3). The rest of this paper discusses each of the aforementioned issues in detail. We seek to identify the architectural reason that causes the difficulties when applying TCP/IP to the IoT world. We also survey the current solutions to those issues that have been standardized or under active development at the IETF, and analyze why they are often insufficient to solve the targeted problems. The goal of this paper is to offer insights and point out directions for the design of future IoT network architectures.

2. Problems at Network Layer

IP, especially IPv6, is engineered for today's Internet environment with desktops and laptops as end devices communicating with wire-connected servers. In this section we discuss which properties of the hosts and the networks currently assumed by IP no longer exist in the IoT world, and what have been done to tailor IP and its companion protocols to fit them into the IoT environment.

2.1 Small MTU

The constrained low-energy links in IoT networks often have very small MTUs. For example, the maximum physical layer frame size for IEEE 802.15.4-2006 [14] is merely 127 bytes. This is in clear contrast with today's IP networks which typically assume a minimum MTU of 1500 bytes or higher. Developed for the traditional Internet during 1990s (long before the perception of IoT), the IPv6 specification [7] includes two design decisions that are problematic for small MTU links. First, IPv6 uses a 40-byte fixed length header with optional extension headers, which cause a big protocol overhead for small packets. Second, the IPv6 specification requires that all IPv6-capable networks support a minimum MTU size of 1280 bytes, which is unrealistic for the constrained links. To fit IPv6 into 802.15.4 networks, 6LoWPAN [19] introduces, between the link layer and the network layer, an adaptation layer that implements two mechanisms to tackle the above mentioned issues: header compression and linklayer fragmentation [13, 20]. Header compression allows the removal of unused fields (e.g., flow label and traffic class) and redundant information (e.g., the interface identifier in the IPv6 address can be derived from L2 MAC address and hence elided). It also defines the compression scheme for extension headers and UDP header, both of which are frequently used in IoT (see Sections 2.4 and 3), in order to leave more room for application payload. Link-layer fragmentation hides the real MTU size of 802.15.4 and gives the network layer the illusion that it is running over a standard compliant link capable of supporting 1280-byte MTU. However, few IoT applications are expected to send packets that reach the MTU limit. The main purpose of having fixed length header in IPv6 is to improve protocol processing speed. Setting a minimum MTU is to avoid in-network fragmentation (which is widely believed to cause performance issues [17]) and reduce the router's workload. Both of them are intended for performance optimization in the current Internet, without the consideration of constrained IoT environment with small MTU sizes. The addition of the adaptation layer patches up the mismatch between the old design and the new usage requirement, which inevitably introduces extra complexity and overhead.

2.2 Multi-link subnet

The current subnet model of IPv4 and IPv6 considers two types of Layer-2 networks: multi-access link, where multiple nodes share the same access medium, and point-to-point link, where there are exactly two nodes on the same link. Both of them assume that the nodes in the same subnet can reach each other within one hop. An IoT mesh network, on the other hand, contains a collection of Layer-2 links joined together without any Layer-3 device (i.e., IP routers) in between. This essentially creates a multi-link subnet model that is not anticipated by the original IP addressing architecture [11]. RFC 4903, "Multi-Link Subnet Issues" [29], documents the reasons why the IETF community decided to abandon the multi-link subnet model in favor of 1:1 mapping between Layer-2 links and IP subnets. The main concerns are around the "one-hop" reachability model that many existing protocols already depend on. First, forwarding across multiple links within the subnet creates trouble with TTL/HopLimit handling. In IP networks it is common practice to limit the scope of communication to a single subnet by setting the TTL/Hop-Limit to 1 or 255 and verify that the value stays the same upon receipt. The multi-link subnet model will break any protocol that follows such practice because the nodes who perform IP forwarding across multiple links will necessarily decrement the TTL/Hop-Limit value. The second issue is that link-scoped multicast does not work on multi-link subnets without proper support for multicast routing (which is often disabled even in today's Internet). Consequently, legacy protocols that depend on link-scoped multicast (e.g., ARP, DHCP, Neighbor Discovery, and many routing protocols) will also be broken on multi-link subnets. Fundamentally, the issues above are caused by the mismatch between the old IP subnet model and the new IoT mesh networks. To avoid those technical issues, one has to either rely on Layer-2 mechanisms to glue multiple links into a single network transparently (similar to bridging of multiple Ethernet segments), or partition the mesh network into multiple subnets with different prefixes. The first approach requires some form of intra-subnet routing capability, which will be discussed in Section 2.4. The second approach introduces new complexity in network configuration as the pre- fix allocation has to be propagated over the mesh network (e.g., via prefix delegation) and the formation of the links in a mesh may change over time in a dynamic environment.

2.3 Multicast efficiency

A lot of IP-based protocols make heavy use of IP multicast to achieve one of the two functionalities: notifying all the members in a group and making a query without knowing exactly whom to ask. However, supporting multicast packet delivery is a big challenge for constrained IoT mesh networks. First, most wireless MAC protocols disable link layer ACK for multicast; consequently lost packets are not recovered at link-layer. Second, multicast recipients may experience different data transmission rate due to the coexistence of multiple MAC protocols (e.g., different versions of Wi-Fi) and/or the link-layer rate adaptation; therefore the sender has to transmit at the lowest common link speed among all receivers. Third, IoT nodes may switch to sleeping mode from time to time to conserve energy, thus may miss some multicast packets. Lastly, when nodes are connected through a mesh network, a multicast packet needs to be forwarded over multiple hops along many paths, potentially waking up many sleeping nodes and overloading the already-scarce network resource. To get around the difficulties in multicast support, the legacy protocols have to be redesigned to minimize the use of IP multicast before they

can be applied to constrained IoT environments. When IoT nodes need to send out notifications to multiple recipients, instead of multicasting the packets, they can buffer those packets temporarily at some well known location and wait for the recipients to pull the packets over unicast on-demand (based on their sleeping schedule). When they want to make queries to a group, instead of flooding the network with multicast, they can send the queries to some designated nodes who are pre-configured to answer queries by collecting the information a prori. These new approaches replace multicast with on-demand unicast pulling, to get around the difficulties in supporting multicast and also to accommodate sleeping nodes. One example of such protocol adaptation is the IPv6 Neighbor Discovery (ND) optimization for 6LoWPAN [24]. The original IPv6 ND [21] relies on multicast to learn default gateway routers, resolve neighbor's IPs to MAC addresses, and perform duplicate address detection. When adapting ND functionalities to 6LoWPAN, instead of having the routers multicast Router Advertisements periodically (which will either wake up the sleeping nodes or be missed by those nodes), the optimized protocol allows the constrained nodes to refresh Router Advertisement information on demand with Router Solicitation messages.1 Another extension is to maintain a registry of host addresses on the routers, making the routers capable of answering address resolution and duplicate address detection requests on behalf of the end hosts, so that the querying nodes simply send their queries to the default routers via unicast messages. An alternative solution called MPL, proposed by the IETF roll WG, fundamentally changes the forwarding semantics of multicast over constrained networks [12]. MPL disseminates multicast packets across the entire multicast domain through synchronization among MPL forwarders (i.e., nodes that participate in MPL) using controlled flooding, without requiring any multicast routing protocol to maintain the topology information. Every multicast packet is identified by the packet generator id and a sequence number in order to allow duplication detection. Also, recent packets are buffered by the MPL forwarders in a sliding-window fashion (i.e., FIFO buffer), which can be used for retransmission in the future. This new multicast forwarding protocol has been adopted by the current ZigBee IP specification [2].

2.4 Mesh network routing

The topologies of typical IoT networks fall into two categories, as is explained in [14]: star topology and peer-topeer (a.k.a., mesh) topology. The routing configuration is straightforward on a star network where the hub node (e.g., a Bluetooth master node) can act as the default gateway for the peripheral nodes. However, the deployment scale of the start topology is limited by the signal coverage of a single hub node, making it unsuitable for application scenarios that cover a wide area. The mesh topology enables larger coverages by having the nodes relay the packets for each 1Note that the Router Solicitation is still a multicast packet, but with an "all-routers" destination address and is only processed by the 6LoWPAN routers. other. Since flooding the whole network is too expensive, a routing mechanism is necessary for implementing efficient packet forwarding inside the mesh. Mesh network routing can be supported at either the link layer or the network layer. The link-layer approach, called mesh-under in the IETF terminology [18], relies on Layer-2 forwarders to join multiple links into a single "one-IP-hop" subnet. The network-layer approach, called route-over, instead relies on IP routers to forward packets across multiple hops. In the rest of this subsection, we describe the existing solution in each of these two categories. The IEEE has produced the 802.15.5 standard [15] to support link-layer routing for mesh networks formed by IEEE 802.15.4 links. The basic approach is to first construct a spanning tree across the mesh network for L2 address assignment: the root of the spanning tree allocates continuous link-layer address blocks to its children, which further allocate sub-blocks to its descendents. Such addressing approach guarantees that the link-layer address of nodes under the same ancestor fall into the same range. Once the addresses are assigned, the nodes start to exchange local link-state information with their immediate neighbors and each of them builds its own 2-hop neighbor table containing the neighbors' address block range, tree level and hop distance. When forwarding packets to a destination beyond 2-hop distance, the sending node applies a simple heuristic to pick a next hop that is close to the spanning tree root (and hence knows more about the network topology) but not too far away from the sending node. One drawback in this solution is that, as new nodes dynamically join the network, the address allocation process may have to be re-performed in order to adapt to the topological changes. The IETF tackles the mesh network routing problem via the route-over approach and has developed RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) [30] as the current standard solution. RPL shares the same spirit with IEEE 802.15.5 in that it models a cluster of nodes as a spanning tree called Destination-Oriented DAGs (DODAG), with all directed paths terminating at the root. When two nodes inside a DODAG communicate with each other, their packets traverse up to either the root node or a common ancestor, then follow a Down Link to the destination. However, unlike IEEE 802.15.5 which allocates topology-dependent L2 address, RPL does not make any assumption about IP address allocation. This effectively prohibits routing entry aggregation beyond the sharing of common prefixes. Maintaining such a routing table becomes quite challenging at the nodes near the root, which in the worst case have to keep routing entries for every device in the subnet. RPL also provides an alternative "Non-Storing" mode, where only the root node maintains the routing table. When forwarding packets along Down Link paths, the root node needs to insert full source route information into the packet headers. While it reduces memory usage on the non-root nodes, the "Non-Storing" mode increases the header size of the downward packets, which is problematic for small-MTU networks (see Section 2.1). We should note that the fundamental challenge of routing in IoT mesh networks comes from the requirement of maintaining routing information for each host in a multilink environment. This is not a issue in traditional IP networks where routers or self-learning bridges can be deployed to provide infrastructural support for routing and forward- ing. However, in constrained IoT environments, the per-host routes are either maintained by every node in the mesh using routing protocols, which consumes lots of memory, or carried with the IP packet as source routes during forwarding, which conflicts with the small MTU restriction

from the link layer. Due to IP's host-oriented communication semantics, routing will remain a major challenge in IP-based IoT mesh technologies.

3. Problems at Transport Layer

The transport layer in the TCP/IP architecture provides congestion control and reliable delivery, both of which are implemented by TCP, the dominant transport layer protocol on the Internet. TCP has been engineered for many years to efficiently deliver a large bulk of data over a long lived point-to-point connection without stringent latency requirement. It models the communication as a byte stream between sender and receiver, and enforces reliable in-order delivery of every single byte in the stream. However, IoT applications usually face a variety of communication patterns which TCP cannot support efficiently. First, due to the energy constraints, devices may frequently go into sleep mode, thus it is infeasible to maintain a long lived connection in IoT applications. Second, a lot of IoT communication involves only a small amount of data, making the overhead of establishing a connection unacceptable. Third, some applications (e.g., device actuation) may have low-latency requirement, which may not tolerate the delay caused by TCP handshaking. When working within lossy wireless networks, the in-order delivery and retransmission mechanism of TCP may also cause head-of-line blocking, which introduces unnecessary delay. Moreover, most wireless MAC protocols also implement link-layer automatic repeat request (ARQ), which may further impair the performance of TCP if the L2 retransmission delay is longer than the TCP RTO [9]. While some industrial IoT standards (e.g., ZigBee IP [2]) still mandate the TCP support, more and more IoT protocols (such as BAC net/IP [1] and CoAP [25]) decided to build transport functionalities into the application layer and chose UDP as the transport layer protocol, which essentially turns the transport layer to a multiplexing module. Such trends highlighted the need for the application level framing [6]. With application level framing, network can identify individual application data units (ADUs), thus enabling more flexible transport support, e.g., apply different retransmission strategies for different types of ADUs, distributing data more efficiently with in-network caching, etc. Unfortunately, current TCP/IP architecture does not allow applications to embed application semantics into network level packets, thus failing to provide sufficient support for application level framing.

4. Problems at Application Layer

Most IoT applications implement the resource-oriented request-response communication model. For example, monitoring applications request data generated by the sensors; and control applications request operations on the physical objects through the actuators. These applications resembles today's Web services that have adopted REST (REpresentational State Transfer) architecture [10] for application-layer communication. Influenced by the huge success of Web, the IoT community has been working on bringing the REST architecture into IoT applications. For example, the IETF core WG has defined "Constrained Application Protocol" (CoAP) standard [25], a UDP-based data transfer protocol customized for constrained environment, to power REST style communication for IoT applications. The need for implementing REST at the application layer highlights the missing support of important functionalities at the lower layers of the TCP/IP architecture, including resource discovery, caching, and security. In this section, we examine how current IoT applications bridge those gaps and the limitation of their solutions.

4.1 Resource discovery

The resource-oriented communication model usually requires a resource discovery mechanism, whereby the applications can request or invoke operations on the resources. The solution for resource discovery in traditional IP networks is DNS-based Service Discovery (DNS-SD) [4]. However, this solution has several limitations in supporting IoT applications. First of all, DNS-SD aims to support service discovery, where the service usually refers to a running program (e.g., a printing service running on some printer). In contrast, the resources in the context of IoT covers a broader scope: besides services, it may also refer to IoT devices, sensor data, etc.. Therefore, the IoT resource discovery requires a more general approach to identify heterogeneous resources. For example, instead of using DNS records, CoAP adopts a URI-based naming scheme to identify the resources (like in HTTP). Based on that, the IETF core WG has developed CoRE-RD [26], a CoAP-based resource discovery mechanism that relies on less constrained resource directory (RD) servers to store the metainfo about the resources hosted on other devices. Secondly, traditional service discovery often relies on multicast when dedicated services such as DNS and CoRE-RD are not available in the local environment. For example, DNS-SD uses Multicast DNS (mDNS) [5] as the carrier of communications for service discovery and name resolution within the local network. However, as we analyzed in Section 2.3, link-local multicast has efficiency issues in IoT environments. An alternative solution to using multicast is to synchronize the resource metainfo across the network in a peer-to-peer fashion (which is similar in spirit to the MPL multicast forwarding protocol we discussed in Section 2.3). For example, the IETF homenet WG is developing the Home Networking Control Protocol (HNCP) [28] to distribute home network configurations using a synchronization mechanism defined by the Distributed Node Consensus Protocol (DNCP) [27]. It is worthwhile to note that the necessity of those solutions is due to the fact that the network and transport layers in TCP/IP are unable to discover the resources defined by the application-layer names. For example, the Neighbor Discovery protocol for IPv6 can only discover configurations at the network layer and below; while the SRV records in DNS-SD typically identify the services by the IP addresses and port numbers. Given the universal demand for resource discovery in the IoT applications, an efficient IoT network architecture should include that as one of its core functionalities and free the applications from implementing their own custom solutions.

4.2 Caching

The TCP/IP communication model requires that both the client (resource requester) and the server (resource holder) are online at the same time. However, in IoT scenarios, the constrained devices may frequently go into sleeping mode for energy saving. Moreover, the dynamic and/or intermittent network environment usually makes it difficult to maintain stable connections between communicating parties. Consequently, the IoT applications often rely on caching and proxying to achieve efficient data dissemination. The selected proxy node can request the resources on behalf of the sleeping nodes and store the response data temporarily until the requesting nodes wake up. The cached contents can also be used to serve similar requests from other nodes who share the same proxy, which saves network bandwidth and reduces response latency. The resource origin server may also appoint some proxy nodes to handle the requests on its behalf (called reverse-proxy) so that it can reduce the client traffic and may go offline when it need to. While it is helpful, the application-level caching implemented by CoAP and HTTP has several limitations in the IoT environment. First, the clients need to explicitly choose a forward- or reverse-proxy node in order to utilize the content caching capability. Those pre-configured caching points may not be optimal for all the client nodes. The clients may utilize the resource discovery mechanism to find nearby proxies on demand. But such solution introduces extra complexity to the whole system. Second, in dynamic network environments where the connectivity is intermittent, the pre-selected proxy point may become totally unreachable. When the network topology changes, the clients need to reconfigure or re-discover the proxies, or otherwise stop using caches and proxies at all. Third, the caches and proxies break the end-to-end connections assumed by the current security protocols (which we will discuss in Section 4.3), making it even harder to protect the application data. To make the caching functionality efficient and flexible in the IoT environment, the network architecture need to provide opportunistic caches pervasively inside the network and allow the applications to utilize them without incurring configuration and communication overhead. This further requires the network layer to be aware of the application layer resources and integrate the caching into the forwarding process so that each network packet can explore the caches as it traverse the network. It also requires a fundamental change to the security model in order to make the in-network caches secure and trustworthy.

4.3 Security

Security is critical to IoT applications due to their close interaction with the physical world. The mainstream security model of IP-based applications is channel-based security (e.g., TLS [8] and its datagram variant DTLS [22]), which provides a secure communication channel between the resource server and the client. The secured-channel solutions, however, do not fit into the IoT environments for several reasons. The first issue with channel-based security is the overhead of establishing a secure channel. Both TLS and DTLS requires two or more rounds of security handshake to authenticate a channel and negotiate the security parameters, before the first application data is sent out. The second issue is that both ends of a channel have to maintain the states of the channel until it is closed. This may impose a high pressure on memory usage when a device needs to communicate with many peers simultaneously in a densely-meshed network. Note that this issue, together with the first one, leads to a difficult tradeoff. The effort of mitigating one issue (e.g., reducing memory usage by establishing short-lived channels on-demand) may deteriorate the other (e.g., each new short-lived channel will have its own handshake overhead). Last but not the least, channel-based security does not guarantee the security of request-response once the application data get out of the channel. This is most troublesome when the middle boxes (e.g., caches and proxies) are deployed to cache the application data. The resource owners need to trust the middle boxes to enforce the access control policies correctly, while the resource requestors need to trust the middle boxes to provide authentic data without tampering. The limitations above highlight the need for a different security model for IoT applications. An alternative model that has been proposed at the IETF is object-based security [23], which secures the application data unit directly rather than the channel through which the data is transmitted. Each data object should carry necessary authentication information (e.g., digital signatures) so that anyone receiving the data can verify its validity regardless of how the data is retrieved. When data confidentiality is the concern, the originator of the data can encrypt the content so that only the intended recipients can decrypt the data. Similar ideas using the object-based security have also appeared outside the IoT area, such as the ongoing efforts at the IETF jose WG to secure JSON objects [3].

5. Rethinking the Architecture

The famous principle of indirection says that “all problems in computer science can be solved by another level of indirection”. But one problem it does not solve is the existence of too many levels of indirection, which precisely describes the situation of the current IoT network architecture. Figure 1 shows the layered structure of an IP-based IoT stack. To support the REST interface, IoT applications usually adopt CoAP or HTTP as the messaging protocol. Usually the applications also need to interact with common services on top of the messaging layer (such as the CoAP Resource Directory and object security support). Right above the transport layer, TLS and DTLS are added to secure the communication channel. In addition, there are multiple infrastructural services that are necessary to facilitate the IP network communications, such as ICMP, DHCP, Neighbor Discovery (ND), DNS and RPL. If we reexamine the network stack by focusing on the core functionalities from the application’s perspective, we will get a rather different picture shown in Figure 2. Instead of “everything over IP”, the IoT applications have converged on a different paradigm of “everything over REST”. At the bottom, an IoT stack may use any data transport such as UDP and 6LoWPAN. In the center of the stack, a RESTful messaging protocol implements all the service components that operate over a single abstraction of the application data unit (ADU) defined by the IoT applications. The contrast between this new

perspective and the layered view of the existing stack reflects the deep-rooted mismatch between the expectations from the IoT applications and the architectural Link Layer (Ethernet/WiFi/Bluetooth/802.15.4/…) with optional adaptation sub-layer IPv6 TCP UDP ICMPv6 DNS/mDNS DHCPv6 ND RPL HTTP CoAP TLS DTLS DNS-SD IoT Apps and Services Figure 1: A typical architecture for IoT systems REST (CoAP/HTTP/…) Data Channel (TCP/UDP/6LoWPAN/…) URI-based Communication Caching Object Security Congestion Control IoT Applications Naming Configuration Discovery Sequencing Reliability Figure 2: An IoT stack from the application's perspective reality of TCP/IP. The REST layer contains several sub-modules that implement critical functionalities: • a URI-based communication mechanism that can deliver application-layer data to network destinations; • a caching mechanism for efficient data dissemination; • an object security mechanism for protecting the integrity and confidentiality of individual ADUs; • a congestion control module that may implement multiple algorithms for different network environments; • naming configuration and resource discovery for assisting the application operations; • a sequencing mechanism for chopping large data that cannot fit into a single ADU; • a reliability mechanism that supports packet retransmission and ordering according to the application's demand. Currently all those functionalities (including the REST interface itself) are implemented by the application layer protocols. However, some of those functionalities could have been more effective if moved into the core network. For example, the congestion control could benefit from the feedbacks of network and link layers to make wiser decisions. Caching could be more efficient if the caches are ubiquitous inside the network, rather than relying on dedicated caching proxies. To utilize in-network caching, URI-based forwarding, REST interface and object security should also be supported at the network layer so that the cached content can be easily located, retrieved and authenticated. This protocol stack optimization eventually lead to a simpler and more efficient architecture that closely resembles the Information Centric Network (ICN) vision. The ICN architectures such as NDN [16, 31] not only provide native support for the functionalities that IoT applications intrinsically demand, but also address the lower-layer network challenges. It applies the same ADU across layers and gives the packet flow control back to the applications. It does not have artificial requirements on minimum MTU; the simplified stack actually reduces the size of packet headers. It is inherently multicast friendly since pervasive caching allows data to be reused by multiple consumers efficiently. Its data oriented communication avoids the issue of addressing and routing to a large number of sensor nodes and opens the opportunity for scalable routing and forwarding over application layer names. The data centric security avoids the overhead entailed by the channel-based security solutions and better suits the IoT devices with limited resources and intermittent connectivity. The architectural simplicity leads to smaller code size for the application software, lower energy and memory footprint for the device, and better utilization of the network resource compared to the current IP-based IoT stack. The potentials of IoT over ICN have already drawn attention at the IRTF icnrg [32] and we expect it to become an active research topic as the interest in the IoT technologies continues to grow.

6. Conclusion

When the TCP/IP protocol stack was first developed in the early 1980s, the goal was to connect mainframe computers through the wired connectivity. Although the protocol stack kept evolving after the IP specification was published, the fundamental assumption behind the architecture design has not changed. IoT networks represent a new type of applications where the IP architecture cannot easily fit in without significant modification to the protocol stack. In this paper, we discussed the challenges of applying TCP/IP to IoT networks that arise from the network and transport layers. We also discussed how the application layer protocols like CoAP provide their own solutions for the desired functionalities that the lower layers fail to support. The mismatch was made more evident by comparing the current IoT stack with the desired architecture from the application's point of view. We proposed an architectural change that moves the REST-related components into the core network layer and eventually arrived at a more efficient architecture to the existing application layer solutions. This new IoT stack would embrace the ICN design and implement the required functionalities natively and more efficiently inside the network.

References

[1] BACnet - A Data Communication Protocol for Building Automation and Control Networks, Mar. 2013.
[2] ZigBee IP Specification Revision 34. ZigBee Document 095023r34, Mar. 2014.
[3] R. Barnes. Use Cases and Requirements for JSON Object Signing and Encryption (JOSE). RFC 7165 (Informational), Apr. 2014.
[4] S. Cheshire and M. Krochmal. DNS-Based Service Discovery. RFC 6763 (Proposed Standard), Feb. 2013.
[5] S. Cheshire and M. Krochmal. Multicast DNS. RFC 6762 (Proposed Standard), Feb. 2013.
[6] D. D. Clark and D. L. Tennenhouse. Architectural Considerations for a New Generation of Protocols. SIGCOMM Comput. Commun. Rev., 20(4):200–208, Aug. 1990.
[7] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Dec. 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112.
[8] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176.
[9] G. Fairhurst and L. Wood. Advice to link designers on link Automatic Repeat reQuest (ARQ). RFC 3366 (Best Current Practice), Aug. 2002.
[10] R. T. Fielding. Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine, 2000.

[11] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), Feb. 2006. Updated by RFCs 5952, 6052, 7136, 7346, 7371
[12] J. Hui and R. Kelsey. Multicast Protocol for Low power and Lossy Networks (MPL). draft-ietf-roll-trickle-mcast-12 (work in progress), June 2015.
[13] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282 (Proposed Standard), Sept. 2011.
[14] IEEE. IEEE Standard for Local and metropolitan area networks–Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). IEEE Std 802.15.4-2006, June 2006.
[15] IEEE. IEEE Recommended Practice for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements Part 15.5: Mesh Topology Capability in Wireless Personal Area Networks (WPANs). IEEE Std 802.15.5-2009, pages 1–166, May 2009.
[16] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.
[17] C. A. Kent and J. C. Mogul. Fragmentation considered harmful. SIGCOMM Comput. Commun. Rev., 25(1):75–87, Jan. 1995.
[18] E. Kim, D. Kaspar, C. Gomez, and C. Bormann. Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing. RFC 6606 (Informational), May 2012.
[19] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational), Aug. 2007.
[20] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), Sept. 2007. Updated by RFCs 6282, 6775.
[21] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), Sept. 2007. Updated by RFCs 5942, 6980, 7048.
[22] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347 (Proposed Standard), Jan. 2012.
[23] G. Selander, J. Mattsson, F. Palombini, and L. Seitz. Object Security for CoAP. draft-selander-ace-object-security-03 (work in progress), Oct. 2015.
[24] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). RFC 6775 (Proposed Standard), Nov. 2012.
[25] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252 (Proposed Standard), June 2014.
[26] Z. Shelby, M. Koster, C. Bormann, and P. van der Stok. CoRE Resource Directory. draft-ietf-core-resource-directory-05 (work in progress), Oct. 2015.
[27] M. Stenberg and S. Barth. Distributed Node Consensus Protocol. draft-ietf-homenet-dncp-12 (work in progress), Nov. 2015.
[28] M. Stenberg, S. Barth, and P. Pfister. Home Networking Control Protocol. draft-ietf-homenet-hncp-10 (work in progress), Nov. 2015.
[29] D. Thaler. Multi-Link Subnet Issues. RFC 4903 (Informational), June 2007.
[30] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), Mar. 2012.
[31] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, kc claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, and E. Yeh. Named Data Networking (NDN) Project. Technical Report NDN-0001, October 2010.
[32] Y. Zhang, D. Raychadhuri, L. Grieco, E. Baccelli, J. Burke, R. Ravindran, and G. Wang. ICN based Architecture for IoT - Requirements and Challenges. draft-zhang-iot-icn-challenges-02 (work in progress), Aug. 2015

Author Profile

T Ramya pursued B.Tech in Electrical and Elctronic Engineering and M.Tech in Computer Networks and Information Security. Currently working in Loyola Academy from 2011.

K ANITHA pursued MCA and M Tech in Computer Science. Currently working in Loyola Academy from 2013.