CHAPTER

Why IP for Smart Objects?

3

In this chapter we argue that IP is the future for smart object networks. There is already a significant momentum for IP-based smart objects as demonstrated by the growing amount of products and systems built upon the principles laid out in this book. In this chapter, we review the challenges inherent to smart object networks, as presented in Chapter 1, and review them in light of the IP architecture discussed in Chapter 2.

Although we advocate the use of the IP architecture and protocols for smart objects, we do not advocate that all smart object networks should be connected to the public Internet. There are some smart objects connected to the Internet, for example, to send data to a central database, but this is an exception, not the norm.

First, a brief recap of the challenges of smart object networks:

- Evolvability: Although we have an idea of where the application space of smart objects is heading, we cannot know what direction it will take in the future. Therefore, smart object technology must inherently support the notion of evolvability. The mechanisms developed for smart objects should not be constrained by today's ideas, but must allow for the next generation of applications to take full advantage of the technology in pursuing its own application goals.
- Scale*:* Smart object networks have a large number of nodes per system. Existing smart object systems have thousands of nodes, and they are likely to develop into systems composed of hundreds of thousands or even millions of nodes. Thus, smart object architecture must support an increasing number of nodes through its addressing, routing, and management mechanisms.
- Diversity of applications: The number of applications for smart objects is large, and so is the number of differences in each application (as seen in Part III). A home automation application does not share all of the properties of an industrial automation application. Smart object technology tailored to one specific application therefore may not work for other applications.
- Diversity of communication technologies: Depending on the application and the environment in which the system is deployed, smart objects can use a wide range of communication technologies. Wireless communication is appropriate in many situations because of its deployment convenience, whereas wired communication is more suitable in other places. Many smart object systems use combinations of disparate technologies in the same deployment.
- Interoperability: Smart object networks need interoperability between the smart object devices and between the smart objects and existing network infrastructures. With the large base of existing systems that smart objects enhance, a smart object architecture that makes interoperability and interconnection difficult or cumbersome will not prevail.

Interconnecting Smart Objects with IP. DOI: 10.1016/B978-0-12-375165-2.00003-X

- Standardization: Mechanisms and protocols that define the operation of smart objects must be standardized using open standards through well-established standardization practices. Any patents covering the standardized technology must be disclosed and made available to be used by third parties. Open standards make the entry barrier low for manufacturers, and allow them to freely choose between different vendors. As seen in Chapter 2, open standards was a key to the success of IP.
- Potentially lossy communication technology: Many of the communication technologies used for smart objects are inherently lossy (data sent are not guaranteed to reach their destinations). Smart object protocols and mechanisms need to take this into account when determining where and how to send data as well as determining when and how often.
- Lifetime: Because of the large-scale installations and demanding applications for smart objects, smart object networks are meant to remain functional for many years. This lifetime has implications both for the performance requirements of smart object mechanisms, which must be power-efficient, and for the mechanisms as such, which must remain operational over the lifetime of the system.
- Low-power consumption: Smart objects have severe power constraints. Many smart objects are powered by batteries that cannot easily be replaced or recharged. Other smart objects draw their energy from their surroundings, such as vibration or electromagnetic energy. In either case, power consumption must be low for the system to achieve its optimal lifetime. The power requirement affects both the network protocols and the construction of nodes. The memory size and computational complexity of the nodes are limited by the power consumption constraints.
- Low cost: Smart objects are deployed in large numbers; therefore a small reduction in per-device costs quickly translates into large savings in the cost of the entire system. Just as the power consumption constraints affect the memory size and computational complexity of the nodes, so do cost constraints. Because of constrained resources such as memory, power, and computation, any smart object architecture must be lightweight.

Given these challenges, we now investigate the IP architecture to find out how well it meets them and their implications.

3.1 INTEROPERABILITY

Interoperability is a predominant characteristic of the IP architecture. It is interoperable because it runs over link layers with very different characteristics, providing interoperability among them (Figure 3.1), and because IP provides interoperability with existing networks, applications, and protocols. We examine these two aspects beginning with how IP provides interoperability between different link layers.

IP was originally designed to provide interoperability at the network layer because it works on top of different types of link layers. A single IP network operates across a variety of underlying media such as Ethernet or WiFi. Within the IP architecture, an IP network operates across both wired and wireless link layers without requiring any external mechanisms or add-ons. Operating over a variety of media has always been the prime objective of the IP architecture.

Interoperability within and across different link layers is very important for smart objects. Smart object networks are composed of a wide variety of link layers and transmission mechanisms. Smart object networks extend from low-power wireless nodes to high-power data coordination

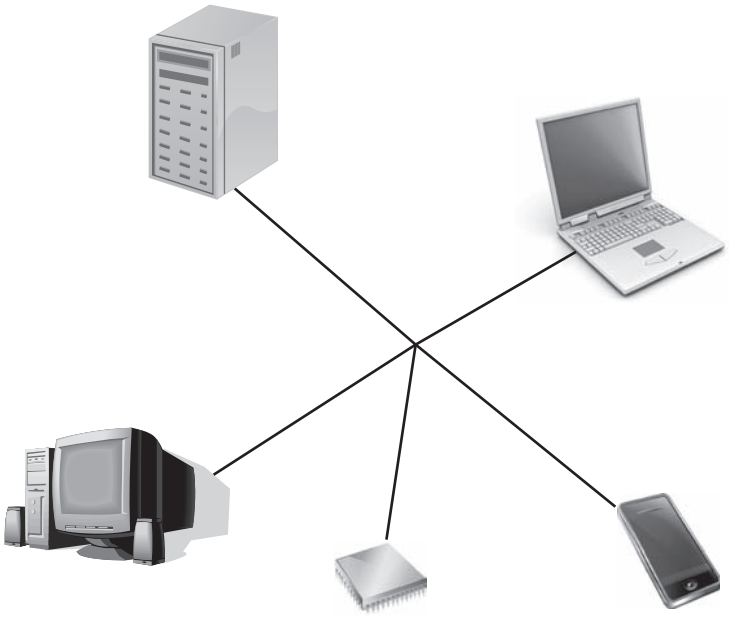

FIGURE 3.1

IP is interoperable across different platforms, devices, and underlying communication mechanisms.

servers. Because of the fundamentally different properties of these devices, it is unlikely they will share a single link layer. A low-power wireless node typically runs a low-power, low-data-rate radio link layer, whereas the high-power data coordination server runs over a wired, high-speed Ethernet network. Still, these systems need to communicate with each other. Because of its layered architecture, IP provides interoperability between these devices without any special servers, gateways, or custom software that connects the systems. IP naturally connects these two. The interoperability of IP is not just an artifact of IP protocols, but occurs because of the architectural choices that support the IP architecture.

The second characteristic of interoperability within the IP architecture is the widespread adoption of IP in today's networked ecosystem. Consequently, an IP-enabled device can interoperate with a large number of devices, computers, and servers. IP is not only the standard protocol that defines the Internet, it is also the de facto standard protocol used for networking computers outside the Internet. IP-based smart objects are able to communicate with any given device without any additional hardware or software.

IP is available in most, if not all, operating systems for general purpose computers and servers, and there is an ever-growing body of software available for IP networking for the type of microcontrollers used in smart objects. Both commercially licensed and open source implementations are generally available: general purpose operating systems such as Microsoft Windows and Linux or microcontroller operating systems such as Contiki, TinyOS, and FreeRTOS. Most software packages also provide the necessary device drivers for the underlying communication hardware.

The ubiquity of IP is also evident in the ever-growing number of communication technologies, or link layers in IP terminology, that support IP. IP runs not only high-speed, high-throughput

communication technology such as the optical links that provide swift communication between servers in data centers, but also low-power, low-data-rate links such as those used for smart objects. This is important for smart object systems designers. With IP, any communication technology the designer chooses will interoperate with other parts of the network infrastructure.

IP-enabled smart objects interoperate with other systems and devices that run IP, but the IP architecture contains other protocols as well. The IP suite contains a set of protocols running on top of IP that include the transport protocols UDP and TCP; application layer protocols such as the Hypertext Transfer Protocol (HTTP), for web-style interaction and web service infrastructure; and the Simple Network Management Protocol (SNMP) for network configuration. Thus a smart object that runs IP is able to interoperate with a large number of external systems.

Interoperability at the application layer is as important for system builders as it is for system integrators. For the system builder, the ability to interoperate with existing application protocols not only makes the act of building the system easier, as existing applications can be used when developing the system, but also when deploying the system. When existing applications are able to interact without any additional mechanisms or heavily tailored software, deployment time is significantly reduced. For the system integrator, system integration becomes much easier when the different parts of the system immediately interoperate with each other.

Standardization plays a large part in the success of IP's interoperability. IP is standardized by an established standardization organization that provides mechanisms through which new standards are reviewed and vetted. This process puts a large amount of effort into ensuring that the mechanisms and protocols proposed as standards can be efficiently implemented. In Part II of this book we describe this process in detail. Furthermore, the standardization body has policies and practices that deal with how patents are to be handled.

3.2 AN EVOLVING AND VERSATILE ARCHITECTURE

The IP architecture has proven to be evolvable due to the way applications, protocols, and mechanisms running on top of the architecture have evolved, and the way that protocols within the architecture have evolved. The ability to evolve and the versatility in applications are due to the end-to-end principle that provides the foundation of the IP architecture.

From the outset the IP architecture was designed to allow application layer protocols and mechanisms to evolve independently of the underlying network protocols and mechanisms. The end-to-end principle states that application layer functionality should be held in the end points of the network (computers, or hosts, connected at the fringes of the network). The network does not contain any application-level intelligence. This is maintained solely by the network end points. The network only transports data between the end points (Figure 3.2).

The network does not know if it is transporting a temperature reading from a temperature sensor, a piece of sound from a voice conversation, a control command, or a piece of a larger file. It only knows that it has been given a string of bits to transport from one end of the network to another. It is up to the applications running at the end points to make sense of the bits.

The end-to-end principle is the primary reason today's IP networks work with a diverse number of applications. If we take the public Internet as an example and look at its history, it shows that the applications running on top of the Internet have evolved since the inception of the Internet in the early

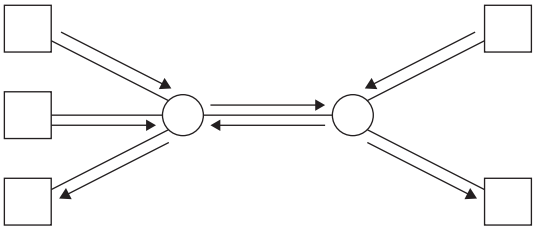

FIGURE 3.2

Versatility is seen when the applications run on the end points and the network only transports data between them, which allows the system to evolve.

1980s. In the 1980s, the Internet was mostly used for transporting text and files; the main applications were e-mail and file transfer between universities. In the 1990s, the World Wide Web was deployed, and by the late 1990s data traffic caused by the Web dominated the traffic on the Internet. In the early 2000s, peer-to-peer file sharing and Internet video transport emerged as new applications, and in 2010 these applications constitute the bulk of Internet traffic.

Without the end-to-end principle, designers might have been impelled to push application functionality into the fabric of the network. For example, the World Wide Web could have been encoded in the routers that make up the interconnected network of the Internet. Placing application functionality within the network may have yielded a slightly higher performance, because data may have needed to travel slightly shorter distances, but evolving the network to support new applications would have been extremely difficult. Inserting a new application into the network would have needed technical cooperation between a large number of parties, and globally agreeing what applications should be supported by the network would have been close to impossible.

In addition to promoting evolvable applications, the end-to-end principle and the resulting architecture embodied in IP have had a profound impact on the interoperability of existing IP networks. If application functionality had been placed deeply in the network fabric, network operators would have needed to negotiate complex deals on how to connect the applications. And once these negotiated deals were in place, adding new applications or evolving new ones would have been difficult.

Thus far we have discussed how the technical architecture that supports IP enables applications running on top of IP to evolve. But there are other elements in the mix that allow the system as a whole to evolve.

We have already touched upon the standards process of IP as an important factor in its interoperability, but the standardization process has implications for the evolution of the architecture too. The well-defined standardization process for IP provides mechanisms through which new features can be introduced to the architecture. The most common example of this is when a new link layer technology is introduced. The standardization process provides a way for vendors to agree on how to use the new link layer to transport IP packets within the IP architecture.

3.3 STABILITY AND UNIVERSALITY OF THE ARCHITECTURE

We have been discussing how the application layer protocols and the underlying link layer mechanisms have allowed IP architecture to evolve. Although evolvability is important, because it shows that the protocols are not tied to one particular application use that may change in the future, stability of the foundations of the architecture is also important. For smart objects, such stability is very important because individual smart object systems are designed to have a long lifetime, often up to ten years.

Such investments require the base technology to be stable enough to remain available toward the end of the system life cycle.

The IP architecture has existed for nearly 30 years. Although there is room in the IP architecture for evolving protocols both at the application layer and at the link layer, throughout the years the architecture as a whole has remained exceptionally stable. Standards have been updated several times over the 30 years, but its foundation as a packet-based communication technology has remained firm. The network layer, the core of the IP architecture, exists in two versions — version four (IPv4) and version six (IPv6). The major difference between the two is that IPv6 provides more addresses. There are, however, no major architectural differences between the two versions.

Because IP forms the basis of the public Internet, the IP architecture and its surrounding standards will continue to exist well into the future. The prevalence of the Internet not only implies that IP has a large installed user base regarding hardware and software that supports it, but there is also a large installed network infrastructure. IP networking equipment and IP network access are both readily available and will continue to be so as long as the Internet exists.

The stability and prevalence of the IP architecture also have implications on the knowledge and education of users and network administrators. IP architecture and its protocols are part of the core curriculum in courses and training material at all levels of the educational system ranging from day-long network training courses to multiyear university programs. Ever year, thousands of new engineers graduate with knowledge of IP protocols and the architecture.

The number of books and training material on IP architecture and its protocols is immense, continues to increase, and is available in many different languages. There is a vast amount of material freely available online both as text, recorded seminars, and animated videos. Again, material is available in many different languages and for different audiences.

3.4 SCALABILITY

The IP architecture has been thoroughly field-proven regarding scalability through the use of IP over the public Internet. Few communication architectures have ever seen such a large-scale deployment. Through the global deployment of the Internet, IP has both shown that it can be deployed over a large number of systems and that it can run across a vast variety of different implementations of its protocols.

But we need not go as far as to the public Internet to witness the scalability of IP. Most larger companies run internal networks to support the activities within the company. These networks are often not connected to the public Internet, yet they can span many thousands of individual computers or servers.

3.5 CONFIGURATION AND MANAGEMENT

Through its wide adoption and large-scale deployment, IP has evolved numerous mechanisms and protocols for network configuration and management. These mechanisms are a necessity when networks grow to thousands of hosts. Network management tools allow for a single person to manage large networks, without manual configuration of each host.

The IP architecture provides advanced configuration and management mechanisms as well as automatic configuration mechanisms. Configuration mechanisms are provided at many layers of the system: from the network layer, where managed and automatic mechanisms for assigning network addresses are widely used, to the routing protocols, where routing mechanisms are both self-healing and automatically configurable.

IP provides management mechanisms at all layers. Address assignment mechanisms such as the Dynamic Host Configuration Protocol (DHCP) allow network administrators to assign addresses both individually to singular nodes and in bulk to others. Routing protocols allow management of both network configuration and engineering.

Protocols such as the widely used SNMP provide means by which a network administrator can inspect the network, its configuration, and its performance. A plethora of tools for interacting with SNMP-enabled networks, and visualizing their performance, exist. The widespread adoption of SNMP also means there is a large body of knowledge and people experienced with these tools. Additional tools such as Cisco Netflow provide large amounts of data about the network health and traffic statistics.

For smart object networks, configuration, management, installation, and commissioning are clearly an issue. Even though traditional management mechanisms cannot be directly applied to smart object networks, due to their large scale and number of nodes, the ability to leverage existing mechanisms and tools is important. It provides not only technical advantages, but also non-technical advantages such as the availability of skilled people.

3.6 SMALL FOOTPRINT

Low energy consumption, small physical size, and low cost are three of the node-level challenges of smart objects. Taken together, these challenges translate into severe memory constraints and software complexity on the nodes. A network architecture for smart objects must be able to run within these tight bounds, and yet perform its task.

The IP architecture was long thought to be a heavyweight due to its perceived need for processing power and memory. The protocols were seen as too large to fit into the constrained environment of typical smart object systems. A typical smart object has only a few tens of kilobytes of memory, whereas existing implementations of the IP protocol family for general purpose computers would need hundreds of kilobytes. For this reason, several non-IP stacks were developed [120,222].

In the early 2000s, however, this view was challenged by lightweight implementations of the IP protocol family for smart objects such as the uIP stack [64]. uIP showed that the IP architecture would fit nicely into the typical constraints of smart objects, without removing any of the essential mechanisms from IP. Note that these resources, which we consider constrained today, are fairly close to the resources of general purpose computers that were available when IP was designed. Since its initial release, the uIP stack has become widely used in networked embedded and smart object systems.

In addition to uIP, there are many small IP stacks available, both as open source and closed source. Many of the early embedded IP stacks were adaptations of the IP stack from the open source BSD UNIX operating system [172].

Recently, a number of implementations of IPv6 for memory-constrained systems have appeared. uIP has been extended to support the IPv6 protocol, which is the first IPv6 stack for smart objects to

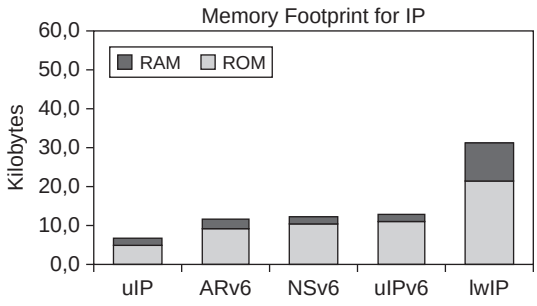

FIGURE 3.3

The memory footprint of uIP, lwIP, and two commercially available IPv6 stacks: the Arch Rock stack (ARv6), and the Sensinode NanoStack (NSv6). The footprint includes transport layer protocols UDP and TCP for uIP and lwIP.

be certified under the IPv6 Ready program [73]. Other independent implementations of the IPv6 stack have also appeared [1,125]. The footprints of the stacks are shown in Figure 3.3. The graph shows the memory requirements of the uIP and uIPv6 stacks [64,73], the stack by Hui and Culler [125], and the lwIP stack [64]. Figure 3.3 shows that there are many options for IP software that fit into the resource constraints in smart object nodes.

In Chapter 13 we take a detailed look at the uIP stack to see how it implements the IP architecture in a way that fits with the smart object resource challenges.

3.7 WHAT ARE THE ALTERNATIVES?

We have now seen that the IP architecture is interoperable across devices and communication technologies, evolving and versatile while still stable, scalable, and manageable, and simple enough that a resource-constrained smart object can easily run it. We have painted a very bright picture of the IP architecture, but is it really as good as we say? What are the alternatives?

The IP architecture was arguably not designed for smart objects. It was designed in the 1970s for connecting general purpose computers using wired networking technologies such as Ethernet. Could we do it better if we made a clean-slate redesign that specifically targets the challenges that smart object networks pose? To help answer our question, we turn to those who did this.

The challenges of low-power operation and the large scale of smart object networks have spurred several years of research in the wireless sensor networks research community. Although wireless sensor networks are a subset of smart object networks, they share many of the properties such as the low-power operation, the large scale of the networks, and the resource constraints.

At the outset, the wireless sensor network community rejected the IP architecture based on the assumption that it would not meet the challenges of wireless sensor network systems [110]. For an emerging research field, this clearly was the right choice. Consequently, many novel network architectures have been investigated, where the layers in the networking stack have been turned upside down [111], where the layers have been intermingled [168], and where the network itself processes the data produced by the end points [162]. After several years, however, the community started to lean toward layered network architectures, because of the benefits of modularity and separation of

concerns [35,46,71,93]. In fact, many have moved to IP because of the interoperability with existing systems and the well-engineered architecture based on the end-to-end architecture [67,73,125,207].

The industry around low-power wireless communication has made a similar transition. In the late 1990s, there was a strong movement toward defining a new network architecture for the networking system under the brand name ZigBee. ZigBee was designed to perform control applications, such as controlling lights and appliances in homes, over a low-power wireless communication medium. ZigBee initially defined a networking stack that would work well over low-power wireless links, but that was incompatible with existing network standards such as IP. In 2009, however, ZigBee announced that they were moving toward adopting IP as its communication mechanism. In Part II of this book, we return to ZigBee to discuss the choices made in the original ZigBee architecture.

Even if we were designing our own network architecture for smart objects, at some point they would need to communicate with someone outside the network. Our electrical meter would need to report its data to a collection server. Our industrial vibration sensor would need to send its latest sensor reading to a database. Our radiator controller would need to be given instructions on how much to turn up the heat in its room. To reach the smart objects, we need to insert a translation point between our smart object network and the outside network. This translation point is called a gateway, and it introduces a number of problems.

3.8 WHY ARE GATEWAYS BAD?

At a first sight, gateways offer an alternative to adopting the IP end-to-end principle, which allowed for interconnecting non-IP-based smart object networks to an IP network.

Such gateways were designed and deployed in a number of networks about a decade ago, when IP was not yet the networking protocol of choice. At that time, several legacy networking protocols such as IBM's Systems Network Architecture (SNA), and Novell's Internetwork Packet Exchange protocol (IPX), and many other ones were deployed mostly in private networks. As IP networks were deployed, network administrators required gateways to interconnect these networks by means of multiprotocol translation gateways supporting these protocols, which led to several deployments models. Some protocols were tunneled over IP (encapsulated in IP packet to transport non-IP traffic over an IP network), while others were translated.

Although such gateways were deployed, most networks very quickly migrated to IP. But why? There are two main reasons for the move away from gateways: the inherent complexity of gateways and the lack of flexibility and scalability.

3.8.1 Inherent Complexity

The mode of operation of a multiprotocol translation gateway is a complex language translation mechanism with subtle nuances in semantics in addition to the actual translation. Network protocol translation is more complex than just a packet format conversion. Networking protocols use different mechanisms and logic for routing, Quality of Service (QoS), error recovery, transport, management, troubleshooting, and security models. Trying to translate the semantics of QoS between two networking protocols, for example, is not limited to the setting of a new field value in a packet and may sometimes not even be possible. Routing is similarly affected: when two routing domains are using

different routing architectures, routing metrics, and paradigms the introduction of protocol translation gateways introduces several limitations. This is true for a number of network aspects where such gateways break the networking models on both sides.

Furthermore, with gateways, management and troubleshooting become cumbersome. Imagine traffic flows between three smart objects implementing different networking protocols. This requires as many as six protocol translations. Such a system is extremely difficult to manage and troubleshoot, especially when the gateway is not managed by a networking expert.

3.8.2 Lack of Flexibility and Scalability

The lack of flexibility and scalability is undoubtedly a real issue. As already pointed out, the evolvability and scalability essential to all networks are required for smart object networks because of the myriad of future innovative applications. Protocol translation gateways inherently do not scale and become networking bottlenecks. Each protocol enhancement implies changes in the gateways, which become the least common denominator factor of the architecture. Furthermore, such gateways introduce an undesirable state in the networks, which impacts not only the overall scalability but also the overall reliability with single points of failure.

The use of multiprotocol gateways helped integrate disparate networks in the late 1990s when network administrators had to deal with several legacy protocols and when networks were significantly smaller. Now that IP has become the networking protocol of choice, the use of multiprotocol translation gateways would ineluctably lead to the wrong architectural choice.

3.9 CONCLUSIONS

Smart object networks and their applications give rise to challenges both at the node and the network level. To meet these challenges we need a network architecture that is interoperable across a wide range of communication technologies, that evolves as the field of smart objects evolves, and is scalable enough to meet the challenges imposed by large-scale smart object networks while lightweight enough for the node-level resource constraints. We argue that the IP architecture meets these goals while providing unprecedented interoperability with existing networks, applications, and services.