Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 21, 2013

H. Tschofenig
J. Arkko
D. Thaler
D. McPherson

March 20, 2013

Architectural Considerations in Smart Object Networking
draft-iab-smart-object-architecture-02.txt

Abstract

Following the theme "Everything that can be connected will be connected", engineers and researchers designing smart object networks need to decide how to achieve this in practice. How can different forms of embedded and constrained devices be interconnected? How can they employ and interact with the currently deployed Internet? This memo discusses smart objects and some of the architectural choices involved in designing smart object networks and protocols that they use.

Status of this Memo

This Internet-Draft will expire on September 21, 2013.

Copyright Notice

Table of Contents

1. Introduction

RFC 6574 [1] refers to smart objects as devices with constraints on energy, bandwidth, memory, size, cost, etc. This is a fuzzy definition, as there is clearly a continuum in device capabilities and there is no hard line to draw between devices that can be classified as smart objects and those that can't.

Following the theme "Everything that can be connected will be connected", engineers and researchers designing smart object networks need to address a number of questions. How can different forms of embedded and constrained devices be interconnected? How can they employ and interact with the currently deployed Internet?

These questions have been discussed at length. For instance, when the Internet Architecture Board (IAB) scheduled a workshop on Smart Objects, the community was asked to develop views on how Internet protocols can be utilized by smart objects. A report of the discussions and the position papers received for this workshop have been published [1].

This memo discusses smart objects and some of the architectural choices involved in designing smart object networks and protocols that they use. The main issues that we focus on are interaction with the Internet, the use of Internet protocols for these applications, models of interoperability, and approach to standardization. Many of the issues discussed in this memo are common to any communications system design or protocol development. However, given the high interest for smart object networks, their somewhat specific requirements, and commonly occurring requests for very different communications tools prompted the IAB to discuss these issues in this specific context.

In drawing conclusions from prior IETF work and from the IAB workshop it is useful to look back at the criteria for success of the Internet. Various publications provide insight into the history, and some of it is very much applicable to the discussion on smart objects. RFC 1958 [2] says:

> "The Internet and its architecture have grown in evolutionary fashion from modest beginnings, rather than from a Grand Plan",

and adds

> "A good analogy for the development of the Internet is that of constantly renewing the individual streets and buildings of a city, rather than razing the city and rebuilding it."

Internet protocols are immediately relevant for any smart object development and deployment. However, building very small, often battery-operated devices is challenging. It is difficult to resist the temptation to build specific solutions tailored to a particular application, or to re-design everything from scratch. Yet, due to network effects, the case for using the Internet Protocol(s) and other generic technology is compelling.

As technology keeps advancing, the constraints that the technology places on devices evolve as well. Microelectronics become more capable as time goes by, sometimes making it even possible for new devices to be both less expensive and more capable than their predecessors. This trend can, however, be in some cases offset by the desire to embed communications technology in even smaller and cheaper objects. But it is important to design communications technology not just for today's constraints, but also tomorrow's.

The rest of the document is organized as follows. Section 2 discusses the problems associated with vertically integrated industry-specific solutions, and motivates the use of generic technologies and a more flexible architecture as a way to reduce these problems. Section 3 discusses the problems associated with attempting to use options and communication patterns other than those in current widespread use in the Internet. Often middleboxes and assumptions built into existing devices makes such usage problematic. Section 4 discusses different levels of interoperability, and the different level of effort required to achieve them. Finally, Section 5 presents some of the relevant security issues, Section 6 discusses privacy, and Section 7 summarizes the recommendations.

2. Specific and General Purpose Solutions

The Internet protocols are relevant for any smart object development and deployment. In the context of one use case of smart objects in particular, RFC 6272 "Internet Protocols for the Smart Grid" [3] identifies a range of IETF protocols that can be utilized.

Of course, there are also many protocols that are unlikely to be needed or even suitable for the smart object environments. For instance, it would difficult to imagine inter-domain routing being a necessary feature in these objects; there are other devices in the network that would normally be responsible for this functionality. But the wide range of protocols listed in RFC 6272 illustrates the view of the IETF about how readily Internet technology can be used in these applications, and indeed Internet communications have been incorporated into various smart object deployments.

Still, many commercial products employ proprietary or industry-specific protocol mechanisms that do not accommodate direct Internet connectivity and researchers have made several attempts to design new architectures for the entire Internet system. There are several architectural concerns that deserve to be highlighted:

Vertically Specified Profiles

The discussions at the IAB workshop (see Section 3.1.2 of [1]) revealed the preference of many participants to develop domain specific profiles that select a minimum subset of protocols needed for a specific operating environment. Various standardization organizations and industry fora are currently engaged in activities of defining their preferred profile(s). Ultimately, however, the number of domains where smart objects can be used is essentially unbounded. There is also an ever-evolving set of protocols and protocol extensions. Profiles, particularly, full-stack profiles are common, for instance, in areas where existing legacy technology is being migrated to IP.

However, merely changing the networking protocol to IP does not necessarily bring the kinds of benefits that industries are looking for in their evolving smart object deployments. In particular, a profile is rigid, and leaves little room for interoperability among slightly differing, or competing technology variations. As an example, layer 1 through 7 type profiles do not account for the possibility that some devices may use other physical media than others, and that in such situations a simple router could still provide an ability to communicate between the parties.

Industry-Specific Solutions

The Internet Protocol suite is more extensive than merely the use of IP. Often significant benefits can be gained from using additional, widely available, generic technologies such as web services. Benefits from using these kinds of tools include access to a large available workforce, software, and education already geared towards employing the technology.

Tight Coupling

Many applications are built around a specific set of servers, devices, and users. However, often the same data and devices could be useful for many purposes, some of which may not be easily identifiable at the time that the devices are deployed.

As a result, the following recommendations can be made. First, while there are some cases where specific solutions are needed, the benefits of general-purpose technology are often compelling, be it choosing IP over some more specific communication mechanism, a widely deployed link-layer (such as wireless LAN) over a more specific one, web technology over application specific protocols, and so on.

However, when employing these technologies it is important to embrace them in their entirety, allowing for the architectural flexibility that is built onto them. As an example, it rarely makes sense to limit communications to on-link or to specific media. We should also design our applications so that the participating devices can easily interact with multiple other applications.

3. Deployment Constraints in the Internet

Despite the applicability of the Internet Protocols for smart objects, picking the specific protocols for a particular use case can be tricky. As the Internet has evolved over time, certain protocols and protocol extensions have become the norm and others have become difficult to use in all circumstances.

Taking into account these constraints is particularly important for smart objects, as there is often a desire to employ specific features to support smart object communication. For instance, from a pure protocol specification perspective some transport protocols may be more desirable than others. These constraints apply both to the use of existing protocols as well as designing new ones on top of the Internet Protocol stack.

The following list illustrates a few of those constraints, but every communication protocol comes with its own challenges.

In 2005, Fonseca, et al. [4] studied the usage of IP options-enabled packets in the Internet and found that overall, approximately half of Internet paths drop packets with options, making extensions using IP options "less ideal" for extending IP.

In 2010, Honda, et al. [5] tested 34 different home gateways regarding their packet dropping policy of UDP, TCP, DCCP, SCTP, ICMP, and various timeout behavior. For example, more than half of the tested devices do not conform to the IETF recommended timeouts for UDP, and for TCP the measured timeouts are highly variable, ranging from less than 4 minutes to longer than 25 hours. For NAT traversal of DCCP and SCTP, the situation is poor. None of the tested devices, for example, allowed establishing a DCCP connection.

In 2011, [6] tested the behavior of networks with regard to various TCP extensions: "From our results we conclude the middleboxes implementing layer 4 functionality are very common -- at least 25% of paths interfered with TCP in some way beyond basic firewalling."

Extending protocols to fulfill new uses and to add new functionality may range from very easy to difficult, as [7] investigates in great detail. A challenge many protocol designers are facing is to ensure incremental deployability and interoperability with incumbent elements in a number of areas. In various cases, the effort it takes to design incrementally deployable protocols has not been taken seriously enough at the outset. RFC 5218 on "What Makes For a Successful Protocol?" [8] defines wildly successful protocols as protocols that are deployed beyond their envisioned use cases.

As these examples illustrate, protocol architects have to take developments in the greater Internet into account, as not all features can be expected to be usable in all environments. For instance, middleboxes [9] complicate the use of extensions in the basic IP protocols and transport-layers.

RFC 1958 [2] considers this aspect and says "... the community believes that the goal is connectivity, the tool is the Internet Protocol, and the intelligence is end to end rather than hidden in the network." This statement is challenged more than ever with the perceived need to develop clever intermediaries interacting with dumb end devices. However, RFC 3724 [10] has this to say about this crucial aspect: "One desirable consequence of the end-to-end principle is protection of innovation. Requiring modification in the network in order to deploy new services is still typically more difficult than modifying end nodes." Even this statement will become challenged, as large numbers of devices are deployed and it indeed might be the case that changing those devices is hard. But RFC 4924 [11] adds that a network that does not filter or transform the data that it carries may be said to be "transparent" or "oblivious" to the content of packets. Networks that provide oblivious transport enable the deployment of new services without requiring changes to the core. It is this flexibility that is perhaps both the Internet's most essential characteristic as well as one of the most important contributors to its success.

4. The Need for Standards

New smart object applications are developed every day; in many cases they are created using standardized Internet technology. Even where a common underlying technology (such as IP) is used, current smart object networks often have challenges related to interoperability of the entire protocol stack, including application behavior. One symptom of such challenges is that various components cannot easily be replaced by third party components. It is of strategic importance to make a conscious decision about the desired level of interoperability and where the points of interconnection are.

4.1. Managing Complexity

These decisions also relate to the required effort to complete the application, and overall complexity of the system. A system may appear complex for variety of reasons. First, there is legitimate heterogeneity in the used networking technology and applications. This variation is necessary and useful, as for instance different applications and environments benefit from varying networking technology. The range and other characteristics of cellular, wireless local area networking, and RFID are very different from each other, for instance. There are literally thousands of different applications, and it is natural that they have differing requirements on what parties need to communicate with each other, what kind of security solutions are appropriate, and other aspects.

The answer to managing complexity in the face of this lies in layers of communication mechanisms and keeping the layers independent, e.g., in the form of the hourglass model. If there is a common waist of the hourglass, then all applications can work over all physical networking technology, ensuring the widest possible coverage of networking applications - ("Everything over IP and IP over everything"). This model provides some guidance for thinking about the Internet of Things architecture. First of all, it shows how we need common internetworking infrastructure (IP) to allow heterogeneous link media to work seamlessly with each other, and with the rest of the system. Secondly, there are various transport and middleware communications mechanisms that will likely become useful in the different applications. For instance, today embedded web services (HTTP, COAP, XML, and JSON) appear to be popular, regardless of what specific link technology they are run over.

But there can also be undesirable complexity and variation. Creation of alternative standards where one would have sufficed may be harmful. Creating systems and communications mechanisms with unnecessary dependencies between different layers and system components limits our ability to migrate systems to the most economic

and efficient platforms, and limits our ability to connect as many objects as possible.

To summarize, complexity and alternative technologies can be very useful as a part of architecture, or can be problematic when it creates unnecessary competition and deployment barriers in the market place. In an optimal situation, complexity will be addressed by regular technological evolution in the industry through underlying layering and modular architecture.

4.2. Interoperability Architecture

It is also valuable to look back at earlier IETF publications, for example, RFC 1263 [12] considers different protocol design strategies and makes an interesting observation about the decision to design new protocols from scratch or to design them in a non-backwards compatible way based on existing protocols:

> "We hope to be able to design and distribute protocols in less time than it takes a standards committee to agree on an acceptable meeting time. This is inevitable because the basic problem with networking is the standardization process. Over the last several years, there has been a push in the research community for lightweight protocols, when in fact what is needed are lightweight standards. Also note that we have not proposed to implement some entirely new set of 'superior' communications protocols, we have simply proposed a system for making necessary changes to the existing protocol suites fast enough to keep up with the underlying change in the network. In fact, the first standards organization that realizes that the primary impediment to standardization is poor logistical support will probably win."

While [12] was written in 1991 when the standardization process in the Internet community was far more lightweight than today (among other reasons, because fewer stakeholders were interested in participating in the standards process) it is remarkable to read these thoughts since they are even more relevant today [13] [14]. This is particularly true for the smart object environment.

Regardless of how hard we work on optimizing the standard process, designing systems in an open and transparent consensus process where many parties participate takes longer than letting individual stakeholders develop their own proprietary solutions. Therefore, it is important to make architectural decisions that keep a good balance between proprietary developments vs. standardized components.

While RFC 1263 [12] certainly provides good food for thought, it also gives recommendations that may not always be appropriate for the

smart object space, such as the preference for a so-called evolutionary protocol design where new versions of the protocols are allowed to be non-backwards compatible and all run independently on the same device. RFC 1263 adds:

> "... the only real disadvantage of protocol evolution is the amount of memory required to run several versions of the same protocol. Fortunately, memory is not the scarcest resource in modern workstations (it may, however, be at a premium in the BSD kernel and its derivatives). Since old versions may rarely if ever be executed, the old versions can be swapped out to disk with little performance loss. Finally, since this cost is explicit, there is a huge incentive to eliminate old protocol versions from the network."

Even though it is common practice today to run many different software applications that have similar functionality (for example, multiple Instant Messaging clients) in parallel it may indeed not be the most preferred approach for smart objects, which may have severe limitations regarding RAM, flash memory, and also power constraints. For example, a smart object that supports only one IP protocol (IPv4 or IPv6) may be preferred over one that supports both, at least from a complexity and cost point of view.

To deal with exactly this problem, profiles have been suggested in many cases. Saying "no" to a new protocol stack that only differs in minor ways may be appropriate but could be interpreted as blocking innovation and, as RFC 1263 [12] describes it nicely, "In the long term, we envision protocols being designed on an application by application basis, without the need for central approval." "Central approval" here refers to the approval process that happens in a respective standards developing organization.

So, how can we embrace rapid innovation with distributed developments and at the same time accomplish a high level of interoperability?

Clearly, standardization of every domain-specific profile will not be the solution. Many domain-specific profiles are optimizations that will be already obsoleted by technological developments (e.g., new protocol developments), new security threats, new stakeholders entering the system or changing needs of existing stakeholders, new business models, changed usage patterns, etc. RFC 1263 [12] states the problem succinctly: "The most important conclusion of this RFC is that protocol change happens and is currently happening at a very respectable clip. We simply propose to explicitly deal with the changes rather keep trying to hold back the flood."

Even worse, different stakeholders that are part of the Internet

milieu have interests that may be adverse to each other, and these parties each vie to favor their particular interests. In [15], Clark, et al. call this process 'the tussle' and ask the important question: "How can we, as designers, build systems with desired characteristics and improve the chances that they come out the way we want?" In an attempt to answer that question, the authors of [15] development a high-level principle, which is not tailored to smart object designs but to Internet protocol develop in general:

> "Design for variation in outcome, so that the outcome can be different in different places, and the tussle takes place within the design, not by distorting or violating it. Do not design so as to dictate the outcome. Rigid designs will be broken; designs that permit variation will flex under pressure and survive."

In order to accomplish this, Clark, et al. suggest to

1. Break complex systems into modular parts.

2. Design for choice.

These are valid guidelines, and many protocols standardized in the IETF have taken exactly this approach, namely to identify building blocks that can be used in a wide variety of deployments. Others then put the building blocks together in a way that suits their needs. There are, however, limits to this approach. Certain building blocks are only useful in a limited set of architectural variants and producing generic building blocks requires a good understanding of the different architectural variants and often limits the ability to optimize. Sometimes the value of an individual building block is hard for others to understand without providing the larger context, which requires at least to illustrate one deployment variant that comes with a specific architectural setup. That said, it is also critical to consider systemic interdependencies between the set of elements that constitute a system, or else they impose constraints that weren't envisioned at the outset.

Since many Internet protocols are used as building blocks by other organizations or in deployments that may have never been envisioned by the original designs, one can argue that this approach has been fairly successful. It may, however, not lead to the level of interoperability many desire: they want interoperability of the entire system rather than interoperability at a specific protocol level. Consequently, an important architectural question arises, namely "What level of interoperability should Internet protocol engineers aim for?"

In the diagrams below, we illustrate a few interoperability scenarios

with different interoperability needs. Note that these are highly simplified versions of what protocol architects are facing, since there are often more parties involved in a sequence of required protocol exchanges, and the entire protocol stack has to be considered - not just a single protocol layer. As such, the required coordination and agreement between the different stakeholders is likely to be far more challenging than illustrated. We do, however, believe that these figures illustrate that the desired level of interoperability needs to be carefully chosen.

4.3. Internet Protocols for Proprietary Protocol Developments

Figure 1 shows a typical deployment of many Internet applications. Here an application service provider (example.com in our illustration) wants to make an HTTP-based protocol interface available to its customers. Example.com allows their customers to upload sensor measurements using a RESTful HTTP design. Customers need to write code for their embedded systems to make use of the HTTP-based protocol interface (and keying material for authentication and authorization of the uploaded data). These applications work with the servers operated by example.com and with nobody else. There is no interoperability with third parties (at the application-layer at least). For instance, Alice, a customer of example.com, cannot use their embedded system which was programmed to use the protocol interface for Example.com with another service provider without re-writing at least parts of her embedded software. Nevertheless, example.com use standardized protocol components to allow for communication across the Internet and for speeding-up the process of software development. This is certainly useful from a time-to-market and cost efficiency point of view. For example, example.com could rely on HTTP, offer JSON to encode sensor data, and use IP to allow various nodes to communicate with each other.

```
            .................
            |  Application  |
            |  Service      |
            |  Provider     |
            |  example.com  |
            |_______________|
                _,   .
              ,'      `.      Proprietary
           _,'          `.    Protocol offered
         ,'               `._ by example.com
       -'                    -
    ,''''''''''''''|       ,''''''''| Sensors
    | Temperature |       | Light  | operated by
    | Sensor      |       | Sensor | customers of
    |.............'       |........' example.com
```

Figure 1: Proprietary Deployment

Clearly, the above scenario does not provide a lot of interoperability even though standardized Internet protocols are used.

Figure 2 shows another scenario. Here example.com is focused on storage of sensor data and not on the actual processing. It offers an HTTP-based protocol interface to others to get access to the uploaded sensor data. In our example, b-example.com and c-example.com are two of such companies that make use of this functionality in order to provide data visualization and data mining computations. Example.com again uses standardized protocols (such as RESTful HTTP design combined with OAuth) for offering access but overall the entire protocol stack is not standardized.

```
                                              .................
                                              |  Application  |
                                             .|  Service      |
                                          ,-` |  Provider     |
                                        .`    | b-example.com |
                                     ,-`      |_______________|
                                   .`
             .................  ,-`
             |  Application  |-` Proprietary
             |  Service      |   Protocol
             |  Provider     |
             |  example.com  |-,
             |_______________|  '.
                  _,              `',
    Proprietary ,'                   '.             ...
    Protocol _,'                       `',    .................
           ,'                             '.  |  Application  |
         -'                                 `'|  Service      |
      ,''''''''|                              |  Provider     |
      | Light  |                              | c-example.com |
      | Sensor |                              |_______________|
      |........'
```

Figure 2: Backend Interworking

4.4. Interoperability

In contrast to the scenario described in Section 4.3, Figure 3 illustrates a sensor where two devices developed by independent manufacturers are desired to interwork. To pick an example from [1], consider a light bulb switch that talks to a light bulb with the requirement that each may be manufactured by a different company, represented as manufacturer A and B.

```
                        _,''''   ''''
                       /     -'``    \
                      |               |
                      \               |
                      /               \
    ,''''''''|       /   Standardized  .       ,''''''''|
    | Light  | ------|---Protocol-------\------| Light  |
    | Bulb   |        .                 |      | Switch |
    |........'         `'-              /      |........'
                          \      _-...-`
    Manufacturer           `. ,.'              Manufacturer
        A                    `                      B
```

Figure 3: Interoperability between two independent devices

In order for this scenario to work manufacturer A, B, and probably many other manufacturers' lightbulbs and light switches need to get together and agree on the protocol stack they would like to use. Let us assume that they do not want any manual configuration by the user to happen and that these devices should work in a typical home network. This consortium needs to make a decision about the following protocol design aspects:

- Which physical layer should be supported?

- Which IP version should be used?

- Which IP address configuration mechanism(s) are integrated into the device?

- Which communication architecture shall be supported? (In [16] Arkko, et al. explain how the complexity of an application heavily depends on the chosen communication architecture and discusses an application with limited communication capabilities, which also translates into low energy consumption.)

- Is there a need for a service discovery mechanism to allow users to discover light bulbs they have in their home or office?

- Which transport-layer protocol is used for conveying the sensor readings/sensor commands? (e.g., UDP)

- Which application-layer protocol is used? (for example, CoAP)

- How are requests encoded? (e.g., as URIs) How is the return data encoded? (e.g., JSON)

- What data model is used for expressing the different light levels? (e.g., [17])

- Finally, some thoughts will have to be spent about the security architecture. This includes questions like: what are the ssecurity threats? What security services need to be provided to deal with the identified threats? Where do the security credentials come from? At what layer(s) in the protocol stack should the security mechanism reside?

This list is not meant to be exhaustive but aims to illustrate that for every usage scenario many design decisions will have to be made in order to accommodate the constrained nature of a specific device in a certain usage scenario. Standardizing such a complete solution

to accomplish a full level of interoperability between two devices manufactured by different vendors will take time.

4.5. Design for Change

With the description in Section 4.3 and in Section 4.4 we present two extreme cases of interoperability. To "design for varation in outcome", as postulated by [15], the design of the system does not need to be cast in stone during the standardization process but may be changed during run-time using software updates.

For many reasons, not only for adding new functionality, it can be said that many smart objects will need a solid software update mechanism. Note that adding new functionality to smart objects may not be possible for certain classes of constrained devices, namely those with severe memory limitations. As such, a certain level of sophistication from the embedded device is assumed in this section.

Software updates are common in operating systems and application programs today. Arguably, the Web today employs a very successful software update mechanism with code being provided by many different parties (i.e., by websites loaded into the browser or by the Web application). While JavaScript (or the proposed successor, Dart) may not be the right choice of software distribution for smart objects, and other languages such as embedded eLua [18] may be more appropriate, the basic idea of offering software distribution mechanisms may present a middleground between the two extreme interoperability scenarios presented in this section.

5. Security Considerations

Section 3.3 of [1] reminds us about the IETF workstyle regarding security:

> In the development of smart object applications, as with any other protocol application solution, security must be considered early in the design process. As such, the recommendations currently provided to IETF protocol architects, such as RFC 3552 [19], and RFC 4101 [20], apply also to the smart object space.

In the IETF, security functionality is incorporated into each protocol as appropriate, to deal with threats that are specific to them. It is extremely unlikely that there is a one-size-fits-all security solution given the large number of choices for the 'right' protocol architecture (particularly at the application-layer). For this purpose, [3] offers a survey of IETF security mechanisms instead of suggesting a preferred one.

A more detailed security discussion can be found in the report from the 'Smart Object Security' workshop [21] that was held prior to the IETF meeting in Paris, March 2012.

6. Privacy Considerations

In 1980, the Organization for Economic Co-operation and Development (OECD) published eight Guidelines on the Protection of Privacy and Trans-Border Flows of Personal Data [22], which are often referred to as Fair Information Practices (FIPs). The FIPs, like other privacy principles, are abstract in their nature and have to be applied to a specific context.

From a technical point of view, many smart object designs are not radically different from other application design. Often, however, the lack of a classical user interface, such as is used on a PC or a phone, that allows users to interact with the devices in a convenient and familiar way creates problems to provide users with information about the data collection, and to offer them the ability to express consent. Furthermore, in some verticals (e.g., smart meter deployments) users are not presented with the choice of voluntarily signing up for the service but deployments are instead mandated through regulation. Therefore, these users have no right to consent; a right that is core to many privacy principles including the FIPs. In other cases, the design is more focused on dealing with privacy at the level of a privacy notice rather than by building privacy into the design of the system, which [23] asks engineers to do.

Similarly, in many applications, smart objects technology is deployed by someone other than the potentially impacted parties. For instance, manufacturers and shops deploy RFID tags in products or governments deploy roadside sensors. In these applications the impacted parties, such as a shopper or car-owner, may not even be aware that such technology is used, and information about the impacted party may be collected.

The interoperability models described in this document highlight that standardized interfaces are not needed in all cases. Depending on the choice of certain underlying technologies, various privacy problems may be inherited by the upper-layer protocols and are therefore difficult to resolve as an afterthought. Many smart objects leave users little ability for enabling privacy-improving configuration changes. Technologies exist that can be applied also to smart objects to involve users in authorization decisions before data sharing takes place.

As a summary, for an Internet protocol architect, the guidelines described in [23] are applicable. For those looking at privacy from a deployment point of view, the following additional guidelines are suggested:

Transparency: The data processing should be completely transparent to the smart object owner, users, and possibly impacted parties. Users and impacted parties must, except in rare exceptional cases, be put in a position to understand what items of personal information concerning them are collected and stored, as well for what purposes they are sought.

Data Quality: Smart objects should only store personal data which are adequate, relevant and not excessive in relation to the purpose(s) for which they are processed. The use of anonymized data should be preferred wherever possible.

Data Access: Before deployment starts, it is necessary to consider who can access the personal data recorded in smart objects and under which conditions, particularly with regard to data subjects, to whom (in principle) full and free access to his/her own data should be recognized. Appropriate and clear procedures should be established in order to allow data subjects to properly exercise their rights. A privacy and data protection impact assessment is considered a useful tool for this analysis.

Data Security: Standardized data security measures to prevent unlawful access, alteration or loss of smart object data need to be defined and universally adopted. Robust cryptographic techniques and proper authentication frameworks should be used to limit the risk of unintended data transfers or harmful attacks. The end-user and impacted parties should be able to verify, in a straight-forward manner, that smart objects are in full compliance with these standards.

7. Summary

Interconnecting smart objects with the Internet creates exciting new use cases and engineers are eager to play with small and constrained devices. With various standardization efforts ongoing and the impression that smart objects require a new Internet Protocol and many new extensions, we would like to provide a cautious warning. We believe that protocol architects are best served by the following high level guidance:

Use Internet protocols

Most, if not all, smart object deployments should employ the Internet protocol suite. The Internet protocols can be applied to almost any environment, and the rest of the suite can be tailored for the specific needs.

The deployed Internet matters

When connecting smart objects to the Internet, take existing deployment into consideration to avoid unpleasant surprises. Assuming an ideal, clean-slate deployments is, in many cases, far too opimistic since already available deployed infrastructure is sticky.

Decide about the level of interoperability

Offering interoperability between every entity in an architecture may be an ideal situation for a standards person but comes with a certain cost. As such, starting with a less ambigious standardization goal may be appropriate, particularly for early deployments.

Don't optimize too early

The constrained nature of smart objects invites engineers to invent each and every technique to optimize protocols for special use cases. While some of these optimizations may be necessary, many of them make the overal design complex and the outcome less usable for the generic use case. Examples of current, useful optimizations include tailoring web services transport mechanisms for smart objects while keeping the overall web services model intact ([24]) or education about good ways to implement IP-based protocol stacks ([25]).

This memo provides also some additional, more detailed suggestions for different audiences. The following recommendations are for the designers of smart object systems:

- Aim for a generic design instead of optimizing too early. Note that some optimizations will only be possible in an architectural context, rather than at the level of an individual protocol.

- We encourage engineers to take existing deployment constraints into consideration to allow for a smooth transition path. This requires a clear understanding of the deployment status and also an analysis of the incentives of the different stakeholders.

- Over time, a wide range of middleboxes have been introduced to the Internet protocol suite. Introducing middleboxes in smart object deployments has been proposed many times but their usage is usually harmful. We recommend carefully investigaing whether new features introduced can be supported without any change to middleboxes. This investigation will likely have to go beyond pure specification work, and may require extensive interoperability testing and a clearly articulated extensiblity story. The guidance in [7] is relevant to this discussion. The added architectural complexity, including security and privacy challenges, has to be a subject of design considerations. Middleboxes are often operated by parties other than the communication endpoints. As such, they introduce additional stakeholders into the architecture that often want to be involved when new features are introduced and as such may slow down the ability to innovate at a high speed.

- The application space has historically seen faster innovation cycles, and separating network-layer from application-layer functionality is therefore recommended. In general, we suggest avoiding standardizing complete protocol stacks. The likelihood that those will be outdated by the time standardization is finished is far too high, particularly with application-layer standards.

- Consider what type of interoperability model is appropriate for the task at hand. An architecture that requires fewer interoperability components often has a faster time to market. Selecting what interfaces are open for interworking between components from different operators and vendors is very important.

These recommendations are for the designers of new protocols or protocol extensions in IETF or elsewhere:

- The Internet Protocol stack has a number of building blocks that have proven useful for many applications. We encourage continuing the development of building blocks that are usable in a number of deployment scenarios.

For the development of new components, the recommendations in [1] provide a good starting point. We do, however, encourage protocol engineers to document the interworking of various protocols in at least one complete system to ensure that the individual parts indeed fit together without creating gaps or conflicts.

For researchers we offer the following suggestions:

- Explore the ability to use mobile code distribution also on smart objects.

- Explore the ability to use mobile code distribution also on smart objects.

- We also propose to conduct ongoing research of the deployment status of various Internet protocols. These investigations provide a snapshot for further interactions with the operator community to ensure that IETF protocols can indeed be deployed in today's Internet and may stimulate discussions on how to deal with unpleasant deployment artifacts.

8. IANA Considerations

This document does not require actions by IANA.

9. Acknowledgements

We would like to thank the participants of the IAB Smart Object workshop for their input to the overall discussion about smart objects.

Furthermore, we would like to thank Jan Holler, Patrick Wetterwald, Atte Lansisalmi, Hannu Flinck, Joel Halpern, and Markku Tuohino for their review comments.

10. Informative References

[1] Tschofenig, H. and J. Arkko, "Report from the Smart Object Workshop", RFC 6574, April 2012.

[2] Carpenter, B., "Architectural Principles of the Internet", RFC 1958, June 1996.

[3] Baker, F. and D. Meyer, "Internet Protocols for the Smart Grid", RFC 6272, June 2011.

[4] Fonseca, R., Porter, G., Katz, R., Shenker, S., and I. Stoica, "IP options are not an option, Technical Report UCB/EECS", 2005.

[5] Eggert, L., "An experimental study of home gateway characteristics, In Proceedings of the '10th annual conference on Internet measurement'", 2010.

[6] Honda, M., Nishida, Y., Greenhalgh, A., Handley, M., and H. Tokuda, "Is it Still Possible to Extend TCP? In Proc. ACM Internet Measurement Conference (IMC), Berlin, Germany", Nov 2011.

[7] Carpenter, B., Aboba, B., and S. Cheshire, "Design Considerations for Protocol Extensions", RFC 6709, September 2012.

[8] Thaler, D. and B. Aboba, "What Makes For a Successful Protocol?", RFC 5218, July 2008.

[9] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.

[10] Kempf, J., Austein, R., and IAB, "The Rise of the Middle and the Future of End-to-End: Reflections on the Evolution of the Internet Architecture", RFC 3724, March 2004.

[11] Aboba, B. and E. Davies, "Reflections on Internet Transparency", RFC 4924, July 2007.

[12] O'Malley, S. and L. Peterson, "TCP Extensions Considered Harmful", RFC 1263, October 1991.

[13] Tschofenig, H., Aboba, B., Peterson, J., and D. McPherson, "Trends in Web Applications and the Implications on Standardization", draft-tschofenig-post-standardization-02 (work in progress), May 2012.

[14] Rosenberg, J., "UDP and TCP as the New Waist of the Internet Hourglass", draft-rosenberg-internet-waist-hourglass-00 (work in progress), February 2008.

[15] Clark, D., Wroslawski, J., Sollins, K., and R. Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet, In Proc. ACM SIGCOMM", 2002.

[16] Arkko, J., Rissanen, H., Loreto, S., Turanyi, Z., and O. Novo, "Implementing Tiny COAP Sensors", draft-arkko-core-sleepy-sensors-01 (work in progress), July 2011.

[17] Jennings, C., Shelby, Z., and J. Arkko, "Media Types for Sensor Markup Language (SENML)", draft-jennings-senml-10 (work in progress), October 2012.

[18] "Embedded Lua Project", 2012.

[19] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[20] Rescorla, E. and IAB, "Writing Protocol Models", RFC 4101, June 2005.

[21] Gilger, J. and H. Tschofenig, "Report from the 'Smart Object Security Workshop', March 23, 2012, Paris, France", draft-gilger-smart-object-security-workshop-01 (work in progress), February 2013.

[22] Organization for Economic Co-operation and Development, "OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data", available at (September 2010) , http://www.oecd.org/EN/document/0,,EN-document-0-nodirectorate-no-24-10255-0,00.html, 1980.

[23] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", draft-iab-privacy-considerations-03 (work in progress), July 2012.

[24] Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-14 (work in progress), March 2013.

[25] Bormann, C., "Guidance for Light-Weight Implementations of the Internet Protocol Suite", draft-bormann-lwig-guidance-01 (work in progress), January 2012.

Authors' Addresses

Hannes Tschofenig
Linnoitustie 6
Espoo 02600
Finland

Phone: +358 (50) 4871445
Email: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at

Jari Arkko
Jorvas 02420
Finland

Email: jari.arkko@piuha.net

Dave Thaler
One Microsoft Way
Redmond, WA 98052
US

Email: dthaler@microsoft.com

Danny McPherson
US

Email: danny@tcb.net