Managing constrained IoT devices

Tobias Kloht
FU Berlin

***Abstract*—The Internet of Things needs a good solution for network management and service/device discovery. This work will elaborate its requirements, introduce existing solutions and try to evaluate which solutions best fit the requirements.**

I. Motivation

The Internet of Things (IoT) requires a change of paradigm in the configuration and management of devices as well as the interaction between devices. At first glance it is obvious that the lack of traditional human interfaces on IoT devices, as well as the amount of devices that need to be managed, makes it impossible to manage these devices via direct human interaction. As all IoT devices require a network interface by definition, a configuration over the network is the obvious solution to this problem. The use of remote access protocols however will not scale for the amount of devices in the IoT. It will therefor be necessary to use specialized application layer protocols for network management.

To enable autoconfiguration, it is furthermore necessary to enable automatic discovery of services. With the shift from traditional infrastrucure-based networks to spontaneous wireless network paradigms in the IoT this becomes inevitable.

In the following I will first develop requirements for network management in the IoT, then introduce the standard network management protocols for traditional networking and emerging protocols specialized for the IoT.

II. Network Management

When managing network infrastructure, administrators are generally facing equipment without human interface devices in remote locations. Apart from that the increasing scale of networks makes it necessary to aggregate administrative tasks on common devices, as opposed to managing each device seperately via remote login. This led to the development of specialized network management protocols providing universal interfaces to these tasks. Further advantages of this approach are the improved capabilities for machine-to-machine communication, or in general the encouragement of automatization over human effort.

As an initial differentiation of requirements, network management can be divided into monitoring and configuration, i.e. reading device parameters and setting parameters to alter the devices mode of operation. Typical use case could be

- monitoring operating temperatures
- setting network addresses programmatically

A finer granularity is defined in the FCAPS model of the ISO Telecommunications Management Network, which defines the following functional areas:

Fault management Denotes the discovery and correction of error conditions. Errors include hardware failure, overload and misconfiguration. In that case the administrator is notified, if possible the error is corrected automatically.

Configuration management Any network interface must be configured for the topology it is used in. In more general terms, configuration management is the process of establishing and maintaining the desired functionality of a device or system. This includes but is not limited to network properties such as addresses and routes.

Accounting management The usage of resources can be tracked to estimate the cost caused by their users. As a further step quotas can be introduced to prevent fraud as well as economical reasons.

Performance management Performance data is evaluated to report deviations from defined thresholds. If within the scope of the respective application, the software might take corrective action to mitigate performance problems. This typically overlaps with the responsibilities of configuration management.

Security management A basic requirement to security management is logging access to network resources. This allows to detect unsolicited access by adversaries and reconstruct which parts of the network have been compromised. Restricting access to network resources is an integral part of managing networks. A responsibility for network management in this context could be the deployment of public keys, to provide access to the owner.

To summarize, network management protocols offer a unified way to cover all aspects required to managing a computer network.

III. Service and Device Discovery

Network management approaches are commonly coupled with protocols for automatic discovery of devices and services. This enables new devices to be added automatically to the pool of managed devices. Clients can use the device's services without initial knowledge about the network topology.

To outline the requirements of service discovery it helps to explain the relationship between devices and services. Computer Networks are formed by a group of computing devices interconnected by a telecommunications network allowing them to exchange data. This underlying infastructure is however mostly transparent to its users, who consume application layer services offered by the network nodes. Service providers are usually implemented as a server component, in which case the consumer of the service takes the role of a client.

The question which services are available on a network is a complex problem because it is not in a fixed state and the number of possible services and service providers can be very large. On the Internet this problem is mitigated by the fact that services are available globally, because the Internet is a global network by definition. The effect is that services are cross-referenced and indexed in searchable databases, which is preferrable in this case because of the vast amount of available services.

The problem is more important when entering a local network - we may have knowledge about networks created by ourselves, but when joining a previously unknown network we usually do not have any knowledge about its infrastructure and the services available. In that case it is desirable that devices advertise the services they offer to new participants.

For example, network printers usually can not be used directly but have to be added by an administrator with knowledge of the network topology. If each network printer available on a local network got advertised to network participants this step of human interference could be skipped.

This is also beneficial on a self-administered network with a small amount of users. Multiple devices per user have become the default and their number will increase in the future. A handoff from mobile phone to tablet and PC is typical in many workflows, therefor it is desirable that all services offered on the network are available to all possible client devices automatically.

Service discovery is also used to enable auto-configuration of network nodes after address configuration and name resolution is completed. Auto-configuration is an essential feature when managing large networks and especially in context of the IoT.

In short, service discovery is needed to avoid configuration when adding new nodes to a network, both clients and service providers.

IV. Requirements of the Internet of Things

When implementing network management in the Internet of Things, the general requirements of the constrained devices it is composed of have to be taken into account. In particular this means that characteristics taken for granted for Internet nodes do not apply in the IoT due to cost- and/or physical constraints, i.e. size, weight and available power and energy [1]. These limitations introduce tight boundaries to the available processing power, memory and storage space on IoT nodes.

To provide some perspective on this, the following is a classification of constrained devices by the IETF regarding memory and storage:

	data size (e.g., RAM)	**code size (e.g., Flash)**
Class 0	$<<$10 KiB	$<<$100 KiB
Class 1	10 KiB	100 KiB
Class 2	50 KiB	250 KiB

TABLE I
Classes of Constrained Devices [1]

It should be obvious that these constraints diverge from traditional Internet nodes to such an extent that it can not be taken for granted for existing protocols and implementations to work without further efforts. It might be assumed that Moore's Law will lead to a relaxation of these constraints, but with regard to the IoT it is assumed that technological advancements will be used to reduce the cost of devices while preserving the current state of computational power. Low device cost is the key inhibitor to a massive deployment of IoT nodes as it can not be worked around by efficient software implementations. In contrast the additional effort for implementation of efficient software scales very well when deploying a large amount of devices.

Another major constraint is the amount of power which can be consumed. While traditional Internet nodes are connected to the power grid continually or in regular intervals, this can not be assumed for IoT devices. Instead many use non-replacable batteries intended to supply power it's entire lifetime. In this context wireless transmissions often consume a significant amount of the available energy, making it necessary to use power-saving strategies to avoid transmissions unless necessary. The IETF classifies these strategies as follows [1]:

Name	**Strategy**	**Ability to Communicate**
P0	Normally-off	Reattach when required
P1	Low-power	Appears connected, perhaps with high latency
P9	Always-on	Always connected

TABLE II
Classes of Energy Limitation [1]

Besides these limitations on the hardware, constrained nodes typically operate on constrained networks with low throughput, high packet loss and highly asymetric link characteristics [1]. These are mostly caused by environmental and media constraints as well as power saving strategies of the involved nodes.

Due to the combination of the aforementioned constraints in the IoT, it can be concluded that connection-oriented protocols such as TCP are not well suited for this environment because of its increased energy consumption and general incompatibility with unreliable connections.

Another potential problem are verbose and in general text-based data formats because they unnecessarily occupy both network bandwidth and device memory. An example for a particularly inappropriate format in the IoT could be XML, as it requires a relatively big overhead to process in addition.

Besides these requirements introduced by the constraints of IoT devices, the use-cases for network management are different from existing applications. The IoT often uses single-purpose devices which will require are reduced set of management interfaces compared to existing multi-purpose internet devices. For example, a frequently used application for network management is monitoring the operating temperature of a device. This might not make sense in a wireless sensor network with the sole purpose of measuring the environment temperature. In general it can be assumed that the number of sensors available for monitoring is much smaller in constrained

devices. With the same rationale the configuration options are more restricted the specialized a device gets. For example it can be assumed that a networked temperature sensor does not require the ability to define routes, as opposed to most network infrastructure hardware.

V. Traditional solutions for Network Management

A. SNMP

The Simple Network Management Protocol (SNMP) was introduced by the IETF in 1988, making it the oldest commonly used network management protocol. It was improved continually with the most current version being SNMPv3 [2] and is the de facto standard protocol for network management. The protocol is designed to be an easily implemented foundation with low overhead for network management, as such it uses UDP for data transmission.

The SNMP specification is comprised of the following three components [3]:

1) A framework for storing the managed information
2) A set of general-purpose management information variables
3) A protocol for exchanging management information between managed devices and administrative entities

Each aspect of the managed system is represented as an object, which is essentially a data variable modeling the respective resource. The collection of objects describing a particular system is referred to as a *Management Information Base* (MIB) [4], [5]. These MIBs are used as a standardized interface for the communication between SNMP agents and management stations, the two roles defined for an environment using SNMP.

Management stations execute applications which monitor and control network devices, they act as an interface for network administrators into the network management system. A *management agent* is the software running on managed devices, executing the requests issued by management stations [6].

SNMP is the protocol linking management station and agent, with the following basic capabilities:

Get: Retrieves the value of an object at the agent
Set: Sets the value of an object at the agent
Trap: Notification from the agent to a management station in case of significant events

Figure 1 illustrates this interaction between management station and agent, while also providing further details on the functionality of SNMP.

Traps can be used for the *trap-directed polling* technique, which means that agents are only polled infrequently after initialization to reduce network traffic. Instead agents report to a management station in case of an unexpected event such as crashes or reboots. In response the management station may use extensive polling to get more information about the error.

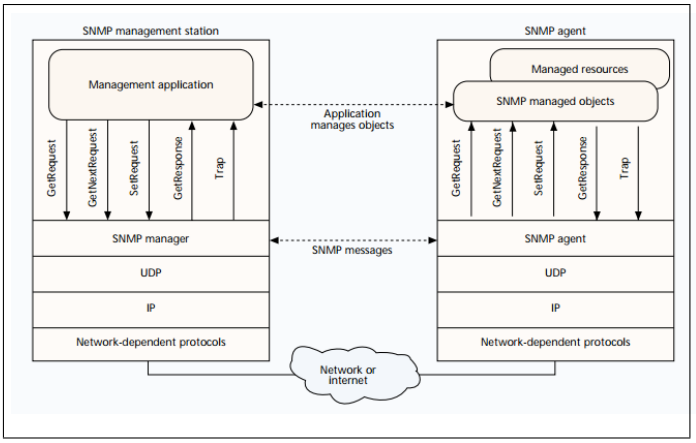

Fig. 1: The interaction between SNMP agent and management station [3]

B. Usage on constrained devices

The wide adoption of SNMP makes it an obvious candidate for IoT network management. This would enable existing tools to manage IoT-nodes without modifications. SNMP was designed to focus on simplicity and minimize the overhead needed to run it, in order to make it usable on as many systems as possible. However, it was not designed for hard constraints as described in section IV, so it needs to be investigated which hardware is the bottom line still able to run SNMP.

The authors of [7] have tried to answer this question by implementing SNMP on the Atmel AVR Raven, a constrained device roughly translating to class 1 as defined in section IV, running Contiki OS [8]. They found that "SNMP makes efficient use of resources on constrained devices" [7], using between six and 24 percent of ROM and less than one percent of RAM, depending on the version of SNMP enabled.

Component	RAM	ROM	Stack
SNMPv1+SNMPv3/USM	235 (1%)	31220 (24%)	1144 (7%)
SNMPv1	43 (0.2%)	8860 (6%)	688 (4%)
NETCONF	627 (4%)	22768 (17%)	678 (4%)
TLS Total	1861 (11%)	37440 (29%)	1834 (11%)
DTLS Total	1961 (12%)	37232 (28%)	2454 (15%)

TABLE III
Comparison of memory usage on an AVR Raven [7]

The same implementation was able to process SNMP requests in 40-120 ms, depending on the used version of SNMP and the chosen security level, as illustrated in figure 2.

It remains unclear though how well SNMP is able to cope with the limitations of constrained networks such as high packet loss and low throughput. Besides that the impact on energy consumption is not evaluated yet, while being a major limitation in the IoT. Further experiments should be conducted to answer these questions.

Another approach to incorporate SNMP in IoT management is the use of a proxy to translate SNMP requests into a protocol designed for usage in constrained environments. This would maintain compatibility with existing management tools using SNMP while making it unnecessary to the problems

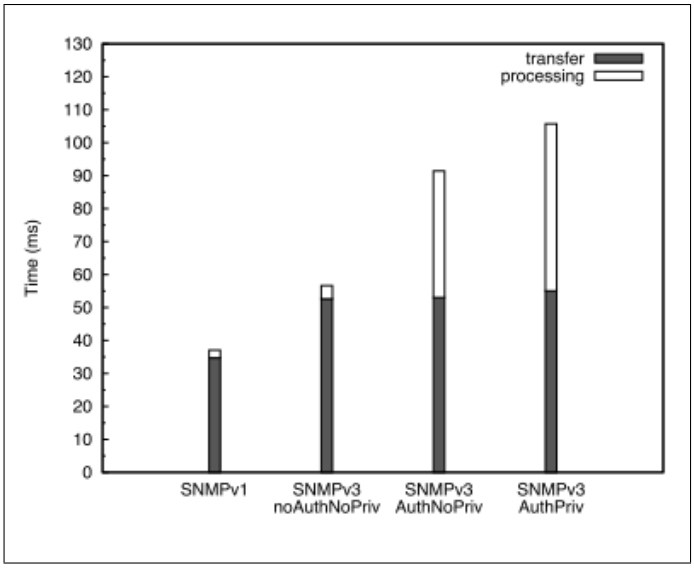

Fig. 2: Average processing and transfer time of SNMP [7]

of its usage on constrained devices. The authors of [9] have developed a prototype for this approach using the Constrained Application Protocol (CoAP) [10] for communication from the constrained network to the gateway. CoAP is an application layer protocol designed by the IETF Constrained RESTful Environments (CoRE) working group. It translates easily to HTTP and uses UDP and 6LoWPAN [11], an adaptation layer for usage of IPv6 on constrained devices. The gateway prototype uses CoAP to retrieve information from a constrained network and provides this as an SNMP agent.

As a conclusion it can be stated that the wide adoption of SNMP makes it worthwile to conduct further research on the feasability of its usage in the IoT. Existing prototypes show that SNMP uses resources efficiently, but it is an open question how it would behave in a real-world scenario.

C. Netconf/Yang

The Network Configuration Protocol (NETCONF) is a network management protocol first published in 2006 [12] and revised in 2011 [13]. The intention was to develop a better protocol for device configuration, after realizing that SNMP is often used for network monitoring only while proprietary interfaces are used for configuration.

Compared to SNMP NETCONF puts less emphasis on simplicity, using XML based encoding and connection-oriented delivery over TCP, with mandatory security mechanisms such as SSH. Instead it aims to provide a feature-complete solution for network management in order to make this unified approach more attractive than proprietary solutions.

Configuration and state data is described in the YANG [14] data modeling language, which is based on XML. NETCONF has the capability to "install, manipulate, and delete the configuration of network devices" [13].

D. Usage on constrained devices

In the experiment referenced in V-B a prototype implementation of NETCONF has been tested on the same device and compared to the SNMP implementation [7]. The results can be seen in table V-B, showing that consumption of RAM and ROM is an order of magnitude higher than SNMP. The processing time of simple requests is significantly increased as well at 500 to 900 ms, which is up to 22 times longer than SNMP. The authors attribute this to the overhead necessary for processing XML, taking 80 to 90 percent of the total processing time for major operations. They note that this could be significantly improved by using an optimized XML library using less write operations.

The second major problem identified by the authors occurs when adding a security layer to NETCONF operations. In this case TLS was used, adding a delay in the order of seconds when starting a session. This problem could be mitigated by using hardware encryption support, however constrained devices usually lack this type of hardware.

NETCONF uses TCP for data transmission, which is not considered a good fit for unreliable, low-throughput wireless transmissions. TCP ensures reliable transmission after establishing a connection, as opposed to UDP which enables a best-effort transmission without setup of an end-to-end connection. The acknowledgement and retransmission required for reliable transmission causes increased load on the network nodes, thus increasing energy consumption and decreasing throughput [15]. Overall TCP causes a significant overhead for constrained devices.

A proposal of RESTful interfaces for data defined in YANG [16] might provide a more resource friendly alternative to the data transfer in NETCONF. The use of compact encoding of RESTful APIs in CoAP would reduce the message size by:

- Avoiding TCP session overhead
- Reducing HTTP verbosity
- Using JSON instead of XML

It should be noted that the impact of security mechanisms remains unchanged, since CoAP uses DTLS for transport-layer security which shares most of the underlying cryptography with TLS.

Another proposal aimed to make an implementation of NETCONF on constrained devices feasible is NETCONF Light (NCL) [17]. The main idea of this draft is reducing complexity via modularization, making it possible for a device to announce that it only implements a subset of NETCONF functionality. This will likely result in reduced memory usage and size of code base.

To summarize, many aspects in the design of NETCONF are not well suited for constrained devices but an effort is made to overcome these difficulties. It remains unclear if these proposed changes will be realized. The general viability of a NETCONF implementation on constrained devices has been shown, but it is an open question how this translates to a real-world scenario. The tradeoff between increased

management capabilities and reliable transmission to efficient use of resources has to be evaluated for IoT networks.

VI. Network Management Approaches for the IoT

Section V has introduced existing network management protocols and their adaptations to constrained devices. The protocols introduced in the following are specifically designed for use in the IoT.

A. CoAp Management Interfaces

The CoAp Management Interfaces (CoMI) draft [18] describes a way to adapt REST-based protocols developed for constrained node networks to SNMP-based network management. CoMI uses CoAP for data transmission, which is designed to easily translate to HTTP in constrained environments, as described in V-B. The data is encoded in the Concise Binary Object Representation [19], a binary data format aimed to allow compact serialization of common data types. The underlying data model of CBOR is based on the JSON data model, thus conversion between these data formats is easy by design.

The rationale to this approach is to prevent adding a second application layer protocol to constrained devices, assuming that CoAP and CBOR implementations are already present. Another obvious advantage is the efficient usage of resources by utilizing protocols and data types designed specifically for this environment, even though SNMP is reasonably efficient in this regard as outlined in section V-B.

The main challenge for CoAP is the mapping of SNMP-based MIBs encoded in SMIv2 [20] to a CBOR representation. One possible pipeline to achieve this is as follows [21]:

1) Conversion from SMIv2 to YANG-structured data
2) Translation to JSON using data-model driven XML to JSON translation [22]
3) Translation from JSON to CBOR is trivial

The authors of [21] note that this approach still requires transfer of the identifiers from the constrained nodes, leading to increased network traffic and the requirement to store those identifiers on the constrained nodes. They propose usage of the numerically structured Object ID defined in SMIv2-based MIBs and further simplification by factoring out common prefixes. In the presented example the unfactored CBOR representation is not significantly smaller than the original encoding, whereas the factored representation is about 70 percent smaller.

A problem for the adoption of CoMI is the relatively early stage of its development. The protocol is in the status of an Internet-Draft at the time of this writing. The working document is in active development, with the latest draft submitted only weeks ago [18]. A published implementation of the protocol is not known to the auhor.

To summarize, CoMI is a promising approach because it maintains compatibility to existing management solutions while using specialized IoT protocols. The used protocols fit relatively well to SNMP and related datastructures. The early stage of development is an inhibitor to widespread adoption at this point.

B. LWM2M

The Lightweight Machine to Machine (LWM2M) Protocol is a service and device management protocol specified by the Open Mobile Alliance (OMA), a standards development organization for the mobile phone industry [23]. An open source implementation of LWM2M is developed at the Eclipse Foundation under the name "Wakaama" [24].

LWM2M utilizes typical IoT technologies such as CoAP for data transmission and DTLS as a security layer. A differentiator to other protocols is the possibility to use SMS in the transport layer instead of TCP. The protocol stack of LWM2M is presented in table IV.

LWM2M Objects	
LWM2M Protocol	
CoAP	
DTLS	
UDP	SMS

TABLE IV
The protocol stack of LWM2M

As implied by its name LWM2M puts an emphasis on machine to machine communication, while using lightweight protocols in order to be compatible to constrained devices. It is divided into three components:

1) The bootstrap server for initial configuration of managed nodes
2) A central server that manages all LWM2M clients
3) Clients which execute operations assigned by the server and report results of these operations back to the server

The interfaces between the components are defined as follows:

Device discovery and registration: The client announces its existence to the server and advertises its capabilities.

Bootstrap: The client receives its initial configuration from the bootstrap server.

Device management and service enablement: The server enables M2M services over this interface. Operations can be send to the client and the response indicates its result.

Information Reporting: Clients can use this interface to report status information to the server. This can be performed periodically or triggered by events.

The main differentiator of this architecture is the integration of service discovery into the management process. This enables automatic configuration as outlined in section III. For example, a new device would use the service discovery and registration interface to announce its existence to the server. In the next step it can receive its initial configuration from the bootstrap server. At this point the device is ready for operation, can report its status to the LWM2M server and receive updated configuration.

The obvious disadvantage of this architecture is the use of a centralized server for management. This requires additional

effort to set up and introduces a single point of failure. Apart from this it would be desirable to allow interaction with multiple management entities instead of binding a constrained device to a centralized authority.

C. Message Queueing Telemetry Transport

Message Queuing Telemetry Transport (MQTT) is a lightweight messaging protocol implementing the publish-subscribe paradigm [25].

Publish-subscribe describes an interaction scheme where messages are not sent directly to the receivers interested in them. Instead, subscribers can express interest in a class of events and will subsequently be notified of any such event. Publishers broadcast their respective information and it is forwarded to all subscribers interested in this class of information [26]. This approach enables a loose coupling of publishers and subscribers, i.e. publishers do not need to maintain any information about their subscribers and vice-versa.

MQTT uses a broker as an intermediary between publisher and subscriber to realize this messaging scheme. Publishers send a *pub* message to the broker, containing the data to be publish alongside the related topic. Subscribers send a *sub* message to the broker containing the topic they are interested in. If the topic of *sub* and *pub* match the broker forwards the *pub* message to the subscriber.

MQTT itself is not designed for use with constrained devices and some of its properties are not well suited for this kind of environment. In particular MQTT assumes a session-oriented, auto-segmenting point-to-point data transport service such as TCP/IP is provided by the network - this may not be the case on IoT nodes. However, MQTT-S is a publish-subscribe protocol based on MQTT developed for usage in wireless sensor networks. MQTT-S does not assume connection-oriented transport and does not rely on message segmentation or in-order delivery. Besides these reduced requirements MQTT-S defines a gateway to MQTT, allowing MQTT-S clients to be accessed like MQTT nodes. Figure 3 illustrates this architecture.

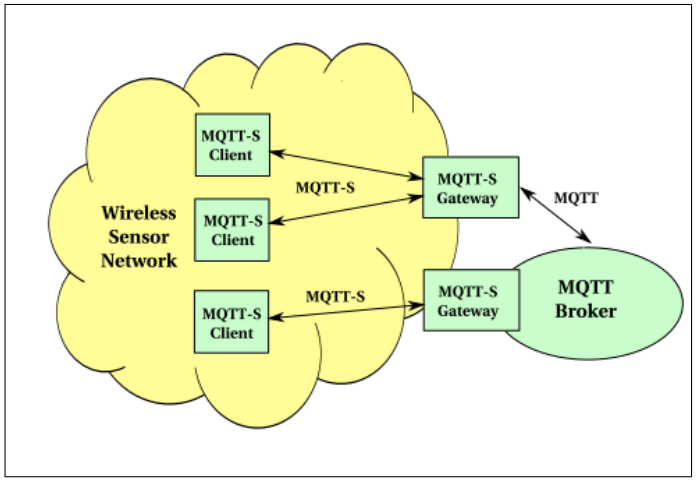

Fig. 3: MQTT-S Architecture [27]

MQTT and MQTT-S is not intended for network management as the previously introduced protocols, instead it provides an efficient way to distribute messages over a network. This can be used to implement network management tasks, an obvious example is the propagation of status information from the publishers. This approach should make significantly more efficient use of network resource compared to polling each node from a network management station.

As another advantage MQTT makes it superfluous to discover or advertise services, because they are managed by the broker. For example a network management station interested in the current battery load of nodes does not require any information about the nodes present in a network. Instead it simply subscribes to the related topic and subsequently receives status updates from all nodes in the network.

A disadvantage can be identified in the need for a broker in addition to the networking nodes, as well as the need for a gateway when using MQTT-S.

Apart from this the protocol lacks the universality of other network management protocols. Nodes have to be configured to use MQTT before deployment, and without further extensions it is not possible to change the mode of operation using MQTT. In comparison, all previously introduced network management solutions incorporate the ability to change the configuration of managed devices and this is an essential requirement of network management.

As a conclusion, MQTT makes efficient use of network resources and could be well suited for specialized tasks in the IoT. On the other hand, it lacks the universality of other protocols introduced in this work.

VII. Service Discovery

As outlined in paragraph III, service discovery is closely related to network management and often used in conjunction. The following paragraph will introduce commonly used protocols for service discovery and their applications to constrained devices.

A. DNS-based service discovery

When implementing service discovery, the general approach is letting a service provider advertise the services it offers. Service consumers then query for a service they want to use and receive a list of the available service providers. The fundamental problem of this idea is the requirement of an intermediary which can be used for advertisement and storage.

Under the assumption that DNS is a bottom-line infrastructure available in all relevant networks, it can be utilized for DNS-based service discovery (DNS-SD) [28]. The procedure is exactly as outlined above - a service provider announces its existence in a special DNS resource record [29]. Consumers can browse this directory to identify consumable service providers and how to access them. To provide a specific use-case for network management, a network node running an SNMP agent could use SNMP-SD to advertise this capability and the available interfaces using DNS-SD [30].

The problem of this solution is its reliance on DNS infrastructure which may not be available in constrained networks.

B. Multicast DNS

Multicast DNS (mDNS) [31] was developed to enable domain resolution in networks lacking a DNS infrastructure. This is implemented by using IP multicast to send DNS-like UDP queries and responses. Names are resolved from local resource records stored by each node.

Multicast DNS is commonly used in conjunction with DNS-SD to enable service discovery without additional DNS infrastructure. This creates a zero-configuration environment where devices can be initialized without previous configuration.

A prototype implementation of mDNS and DNS-SD on constrained devices [32] proves that these protocols are generally usable in an IoT context.

VIII. Conclusion

This work has developed the requirements of network management in the Internet of Things and introduced existing solutions to evaluate their satisfaction of these requirements.

In general it can be stated that each of the presented protocols incorporates advantages and shortcomings compared to each other. At the same time none of the protocols can be seen as the single solution to solve all network management tasks in the internet of things.

One realization of this work is that existing protocols which are widely used outside of the IoT, namely SNMP and NETCONF, can be adapted to constrained devices. The advantage of using existing tools and systems for IoT management should be reason enough to consider further usage of these protocols.

A special case in this regard is CoMI because it allows interoperability with existing SNMP infrastructure, thus the same argument can be made for it.

DNS based service discovery is an important enhancement to these protocols and existing research suggests that it can be adopted to constrained devices.

The remaining protocols MQTT and LWM2M are less ubiquitous in comparison but serve their respective niche very well.

References

[1] M. Ersue and C. Bormann, "Terminology for Constrained Node Networks." [Online]. Available: http://tools.ietf.org/html/draft-bormann-lwig-terms-00

[2] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," RFC 3411 (INTERNET STANDARD), Internet Engineering Task Force, Dec. 2002, updated by RFCs 5343, 5590. [Online]. Available: http://www.ietf.org/rfc/rfc3411.txt

[3] W. Stallings, "SNMP and SNMPv2: The infrastructure for network management," *Communications Magazine, IEEE*, 1998. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=663326

[4] R. Presuhn, "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)," RFC 3418 (INTERNET STANDARD), Internet Engineering Task Force, Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3418.txt

[5] K. McCloghrie and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets:MIB-II," RFC 1213 (INTERNET STANDARD), Internet Engineering Task Force, Mar. 1991, updated by RFCs 2011, 2012, 2013. [Online]. Available: http://www.ietf.org/rfc/rfc1213.txt

[6] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "Simple Network Management Protocol," RFC 1067, Internet Engineering Task Force, Aug. 1988, obsoleted by RFC 1098. [Online]. Available: http://www.ietf.org/rfc/rfc1067.txt

[7] A. Sehgal and V. Perelman, "Management of resource constrained devices in the internet of things," *Communications ...*, 2012. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6384464

[8] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," *Local Computer Networks, ...*, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1367266

[9] H. Lindholm-Ventola and B. Silverajan, "CoAP-SNMP Interworking in IoT Scenarios," Jan. 2014. [Online]. Available: http://trip.cc.tut.fi/dpub/handle/123456789/21993

[10] Z. Shelby, Sensinode, K. Hartke, C. Bormann, and U. B. TZI, "Constrained Application Protocol (CoAP)," 2013. [Online]. Available: http://www.ietf.org/id/draft-ietf-core-coap-18.txt

[11] 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), Internet Engineering Task Force, Aug. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4919.txt

[12] R. Enns, "NETCONF Configuration Protocol," RFC 4741 (Proposed Standard), Internet Engineering Task Force, Dec. 2006, obsoleted by RFC 6241. [Online]. Available: http://www.ietf.org/rfc/rfc4741.txt

[13] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," RFC 6241 (Proposed Standard), Internet Engineering Task Force, Jun. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6241.txt

[14] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020 (Proposed Standard), Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc6020.txt

[15] A. Dunkels, J. Alonso, and T. Voigt, "Making TCP/IP viable for wireless sensor networks," *SICS Research Report*, 2003. [Online]. Available: http://eprints.sics.se/2333

[16] A. Bierman and M. Bjorklund, "YANG-API Protocol." [Online]. Available: http://tools.ietf.org/html/draft-bierman-netconf-yang-api-01

[17] V. Perelman, M. Ersue, J. Schönwälder, and K. Watsen, "Network Configuration Protocol Light (NETCONF Light)." [Online]. Available: http://tools.ietf.org/html/draft-schoenw-netconf-light-01

[18] P. van der Stok and B. Greevenbosch, "CoAp Management Interfaces," 2014.

[19] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," RFC 7049 (Proposed Standard), Internet Engineering Task Force, Oct. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7049.txt

[20] K. McCloghrie, D. Perkins, and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)," RFC 2578 (INTERNET STANDARD), Internet Engineering Task Force, Apr. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2578.txt

[21] O. Bergmann, C. Bormann, and S. Gerdes, "REST-based Access to SMIv2-structured Information on Constrained Devices," *tzi.de*. [Online]. Available: http://www.tzi.de/~cabo/noms2014.pdf

[22] L. Lhotka, "Mapping YANG to Document Schema Definition Languages and Validating NETCONF Content," RFC 6110 (Proposed Standard), Internet Engineering Task Force, Feb. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6110.txt

[23] "LWM2M M2M The Lighter Way — Y.I Readings." [Online]. Available: http://yucianga.info/?p=786

[24] M. O. David Navarro, Julien Vermillard, Manuel Sangoï, Benjamin Cabé, "Wakaama." [Online]. Available: http://www.eclipse.org/proposals/technology.liblwm2m/

[25] "MQ Telemetry Transport (MQTT) V3.1 Protocol Specification," Aug. 2010. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html

[26] P. Eugster and P. Felber, "The many faces of publish/subscribe," *ACM Computing ...*, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=857078

[27] U. Hunkeler, "MQTT-SA publish/subscribe protocol for Wireless Sensor Networks," *... Systems Software and ...*, 2008. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4554519

[28] S. Cheshire and M. Krochmal, "DNS-Based Service Discovery," RFC

6763 (Proposed Standard), Internet Engineering Task Force, Feb. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6763.txt

[29] A. Gulbrandsen, P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," RFC 2782 (Proposed Standard), Internet Engineering Task Force, Feb. 2000, updated by RFC 6335. [Online]. Available: http://www.ietf.org/rfc/rfc2782.txt

[30] T. Tsou, J. Schönwälder, and C. Zhou, "DNS SRV Resource Records for Network Management Protocols." [Online]. Available: http://tools.ietf.org/html/draft-schoenw-opsawg-nm-srv-03

[31] S. Cheshire and M. Krochmal, "Multicast DNS," RFC 6762 (Proposed Standard), Internet Engineering Task Force, Feb. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6762.txt

[32] A. Siljanovski, A. Sehgal, and J. Schönwälder, "Service Discovery in Resource Constrained Networks using Multicast DNS," *anujsehgal.com*. [Online]. Available: https://www.anujsehgal.com/publications/pdf/preprint/siljanovski_eucnc2014.pdf