AgACCP – A Protocol Agnostic API for Content Consumers and Producers

Martine S. Lenders, advised by Matthias Wählisch
Freie Universität Berlin
{m.lenders, m.waehlisch}@fu-berlin.de

***Abstract*—Most protocols in the Internet of Things (IoT) such as CoAP and MQTT follow the end-to-end paradigm of the Internet. NDN on the other hand implements a fundamental change by bringing application layer notion to the networking layer. In consequence application developers are now challenged (not only in the IoT) with the fact that – in addition to the traditional choices like reliability or security – they have to choose whether they need to find a host first to get data or not. The protocol agnostic API for content consumers and producers (*AgACCP*) offers some help with this decision, as it abstracts away any networking related contexts and purely focuses on content production and consumption.**

I. Introduction

The Internet is based on an end-to-end paradigm due to the nature of content exchange in its early days. Today however the Internet is used primarily to disseminate content mostly identified by a name. Especially, the host of a producer lost its importance and often even poses a problem, as the locality to a certain host can lead to a conflict in resource scalability.

Coming from an Internet of Things background, there are three protocols to consider for fetching content: *CoAP* [1] as a GET/Response-based protocol designed by the IETF to allow implementation of HTTP-like RESTful APIs, with the distinction of being usable over non-reliable transport layers like UDP. As such it uses HTTP-like URIs such as `coaps://inf.fu-berlin.de/lecture-hall/temp` to identify content objects. However, at least to date it is not widely deployed. *MQTT* [2] on the other hand is deployed by many cloud providers servicing the IoT domain. It uses a topic-based publish / subscribe paradigm to deliver content: A client connects to a central broker g. `inf.fu-berlin.de`) and depending on their role they either subscribe to topics (e.g. `/lecture-hall/#` using wildcards) to fetch the content (subscribers) or publish data under a certain topic (publishers; e.g. `/lecture-hall/temp/`).

Finally, NDN is an implementation of Information-Centric Networking [3]. It implements the publish / subscribe paradigm on the network layer. It uses hierarchical naming for the content to make routing decisions and is also able to provide on-route caching, since the content and the name of the content are strongly bound. A name example would be `/de/fu-berlin/inf/lecture-hall/temp`.

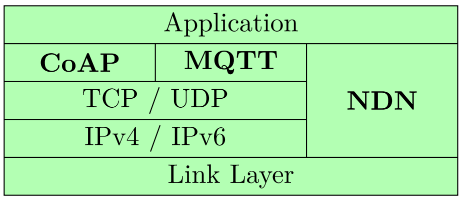

Fig. 1. Network stack utilizing CoAP, MQTT, and NDN.

Using all three protocols at once would result in a network stack seen in Figure 1.

While all three protocols provide a defined API for developers to interact with, a common API for all three would be desirable to minimize porting efforts and to provide gateways between the three domains of the protocols.

The API proposed here – *AgACCP* ([əˈgaːsp]) – is providing such a common development interface.

II. Challenges

A number of challenges arise when implementing such a common API.

First, there is a semantic difference with regards to naming to an MQTT topic which is comparable to requesting all content objects published under this topic. In contrast to this CoAP and NDN identify a single content object by name. A well defined name translation between those semantics would thus be required.

Second, the question of duplicate names arise. There might be a content object published, e.g., in both the

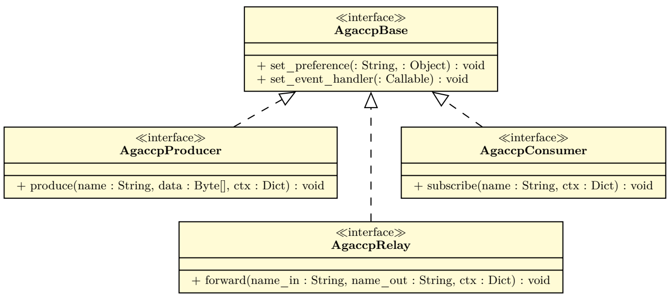
≪interface≫
AgaccpBase
+ set_preference(: String, : Object) : void
+ set_event_handler(: Callable) : void
≪interface≫
AgaccpProducer
+ produce(name : String, data : Byte[], ctx : Dict) : void
≪interface≫
AgaccpConsumer
+ subscribe(name : String, ctx : Dict) : void
≪interface≫
AgaccpRelay
+ forward(name_in : String, name_out : String, ctx : Dict) : void

Fig. 2. Class model for AgACCP

MQTT broker and an NDN / CoAP network. One alternative would be to enforce a syntactic different between names and thus prevent the occurrence of duplicate names. Due to the agnostic nature of *AgACCP* this is not desirable. The second approach would need to address the problem of redundant content. Two solutions might a heuristic (for the embedded context less desirable) or a preference via user configuration.

Third, from a systems perspective, a constrained IoT node is equipped with a modular architecture in which different network stacks share common data structures where possible to allow for a minimal, use-case dependent code size.

III. AgACCP

AgACCP provides an **ag**nostic **A**PI for **c**ontent **c**onsumption and **p**roduction, be it ICN-based communication, traditional HTTP over TCP/IP, or IoT protocols such as MQTT(-SN) or CoAP. It is designed with versatility and use on constrained devices in mind.

When developing an API for constrained devices, experience shows that it should be as minimal as possible and also allow for only implementing sub-sets of the API's operations depending on the use case. As such, the roles of producers and subscribers should clearly be distinct from each other. To bridge domains that deploy different technologies, a relay function is needed.

Based on those observations, *AgACCP* was developed. A class model can be seen in Figure 2.

A base interface `AgaccpBase` provides a way to both configure the end-points preferences and to set an event handler. The event handler is a callback that can be used by the underlying implementation to notify about events such as finished content production or the retrieval of a content a consumer subscribed to.

Producers and consumers extend the `AgaccpBase` interface as `AgaccpProducer` and `AgaccpConsumer` respectively. Both provide a single method: `AgaccpProducer::produce()` can be used to produce content `data` under a certain `name` and `AgaccpConsumer::subscribe()` in turn allows for subscription to a certain `name` which consumption is notified using the event handler. In both cases, the `ctx` dictionary allows the user to set some protocol specific contexts, if required by the user (e.g. protocol preference for duplicate names).

To allow bridging two domains (e.g. to implement an ICN-to-MQTT-gateway) a third interface `AgaccpRelay` that also extends `AgaccpBase` is specified. Its `AgaccpRelay::forward()` method allows the user to define a rule how to transform one name (`name_in`) into another (`name_out`).

A. Related work

An agnostic API for the transport layer called Transport Services is currently worked on by the TAPS working group of the IETF [4]. AgACCP is orthogonal to this, since it is targeted to be an API to application layer protocols, while Transport Services is target for the transport layer. Furthermore, AgACCP was designed with protocols that are not working end-to-end while such protocols can only be implemented rather awkwardly using Transport Services. However, Transport Services (TAPS) could be used to implement *AgACCP* over the transport protocols supported by TAPS such as HTTP.

A similar approach more focused on providing an API to NDN was presented in [5].

IV. Next Steps

A first step would be an implementation that is supported by a few platforms and supports a good subset of underlying protocols. Both ends of the spectrum would be desired for a good comparability. This means both a high-end implementation in a high-level language such as Python and an implementation for constraint devices in a low-level language such as C.

References

[1] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," IETF, RFC 7252, June 2014.

[2] A. Banks and R. G. (Eds.), "MQTT Version 3.1.1," OASIS, OASIS Standard, October 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[3] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[4] T. Pauly, B. Trammell, A. Brunstrom, G. Fairhurst, C. Perkins, P. Tiesel, and C. Wood, "An Architecture for Transport Services," IETF, Internet-Draft – work in progress 02, October 2018.

[5] I. Moiseenko, L. Wang, and L. Zhang, "Consumer/producer communication with application level framing in named data networking," in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*. ACM, 2015, pp. 99–108.