Workgroup: CoRE
Internet-Draft:
draft-amsuess-core-coap-kitchensink-03
Published: 13 March 2023
Intended Status: Informational
Expires: 14 September 2023
Authors: C. Amsüss

Everything over CoAP

Abstract

The Constrained Application Protocol (CoAP) has become the base of applications both inside of the constrained devices space it originally aimed for and outside. This document gives an overview of applications that are, can, may, and would better not be implemented on top of CoAP.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (core@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/core/.

Source for this draft and an issue tracker can be found at https://gitlab.com/chrysn/coap-kitchensink.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 September 2023.

Copyright Notice

Table of Contents

1. Introduction

[See abstract for now]

2. Applications

2.1. Publish-Subscribe services

Publish-subscribe services (pubsub) are a widespread tool for the some of the fundamental use cases of Internet of Things (IoT) protocols: acquiring sensor data and controlling actuators.

A pubsub implementation has been in development since shorlty after the original CoAP publication and is as of now still in draft status, as [I-D.ietf-core-coap-pubsub].

Competing with
MQTT

Strong points Once a topic is set up, data can be sent and received by CoAP clients that are not even aware of pubsub, as long as they can PUT or GET (possibly with observation) data to and from configured URIs.

Weak points To implement a pubsub broker that supports arbitrarily many topics, some (potentially difficult-to-implement) compromises have to be made.

2.2. Remote configuration

The OMA LwM2M protocol (which caters for several applications at the granularity of this document) includes provisions for configuring and monitoring devices over the network, setting properties such as a time server and reading properties such as a network interface's packet count.

In parallel, the NETCONF protocol and its YANG modelling language have been ported to the constrained ecosystem as CORECONF [I-D.ietf-core-comi]. By using numeric identifiers with good compression properties, it can efficiently express data both from shared and from bespoke models in single requests.

Competing with SNMP [?], Puppet [?]

2.2.1. Network status monitoring

Related to remote configuration, CoAP is used as the signalling channel of DOTS ([RFC132]).

Strong points CoAP over UDP/DTLS provides operational signalling on links under attack, on which a TCP/TLS based connection would fail.

CoAP's consistency across transports makes it easy to adjust to situations in which UDP is uanvailable, sacrificing some properties but leaving the high-level protocol unmodified.

Weak points CoAP's default parameters for flow control (such as PROBING_RATE) are unsuitable for this application and need to be customized.

2.2.2. Runtime configuration

Related to remote network configuration, but used without human intervention, CoAP is used negotiate cryptographic keys and other short-lived network configuration, eg. in [CoJP],

[ace-key-groupcomm-oscore], and [OpenThread_Multicast]. This is comparable to how [DHCP] or [IGMP] exchanges configuration.

Strong points When the exchanged communication is essential to joining the network in the first place, CoAP traffic exchanged over a temporary link can easily be proxied into the actual network, even when routing is not an option yet.

2.3. Software updates

The SUIT manifest format [I-D.ietf-suit-manifest] can be used to describe firmware updates that can be performed over CoAP or any other protocol that is expressible in terms of URIs.

The OMA LwM2M protocol also contains provisions for firmware updates over CoAP.

2.4. Network file system

Using CoAP as a backend for a no-frills file service is a simple composition and is provided as a demo by the aiocoap library and a module in the RIOT operating system.

It has never been specified and described; that gap is closed in Appendix A.

Competing with WebDAV, NFS, FTP

Strong points CoAP protocol already provides random read access (through the Block2 option), optimistic locking and cache (through the ETag and If-Match options) and change notification (through the Observe option).

Files can be used in other CoAP protocols without the client's awareness (e.g. for SUIT)

Weak points Transfer of large files is inefficient due to the repetition of file names in block-wise requests (mitigated when using CoAP-over-TCP and BERT).

Advanced file system functionality (file metadata, server-to-server copies, renaming, locking) would need additional specifications.

2.5. Network address resolution

The Domain Name System (DNS) can be utilized from CoAP using the mechanisms described in [I-D.draft-lenders-dns-over-coap].

Strong points

Savings in firmware complexity by using infrastructure shared with other applications.

Potential for traffic (and thus energy) reduction by using request-response binding.

Weak points Not deployed in existing networks.

2.6. Time service

Whereas no attempt has been made yet to specify a time service over CoAP, a primitive time service can be assembled by creating a CoAP resource that returns the server's current time, e.g., in a UNIX time stamp represented in decimal ASCII, or in CBOR using any of timestamp tags (0, 1 or 1001).

Such services have been in use since at least 2013, and are easy to operate and scale.

Competing with SNTP, NTP

Strong points Savings in firmware complexity by using infrastructure shared with other applications.

Compact messages.

Reuse of existing security associations.

Weak points None of the advanced features of (S)NTP, such as distinction between receive and transmit timestamps. Not even leap seconds are advertised (but that can be mitigated by using a time scale that is not affected by them, such as TAI).

Generally only suitable for the last hop in time synchronization.

2.7. Terminal access

Virtual terminal access was one of the first network applications described in an RFC ([RFC15]), and popular to date.

There is no full specification yet as to how to express the data streams of character based user input and character based text responses in CoAP. Necessary components, as well as optional future extensions, have been sketched and implemented for the RIOT operating system at https://forum.riot-os.org/t/coap-remote-shell/3340/5. Unlike SSH, that sketch assumes the presence of a single virtual terminal (as opposed to one created per connection). On platforms with dynamic resources and per-process output capture, an SSH-like muliplexing can be created based on the resource collection pattern described in [I-D.ietf-core-interfaces].

Competing with
SSH

Strong points The head-of-line blocking that occasionally plagues TCP based connections is eliminiated in favor of on-demand recovery (i.e., observing the last output will produce the latest chunk of output, and the terminal may recover skipped data later if it is still in the device's back-scroll buffer).

Weak points The default retransmission characteristics of CoAP make operations painfully slow when encountering packet loss. Tuning of parameters or the implementation of advanced flow control as described in [I-D.ietf-core-fasor] are necessary for smooth operation.

On-demand recovery is incompatible with regular terminals, and requires either fully managed terminals (where the full output is reprinted when lost fragments are recovered) or accepting the loss of data where printed exceeding the network speed. (Data is still lost gracefully, as the loss is detected and can be indicated visually).

2.8. Chat services

The CoMatrix project https://comatrix.eu/ has demonstrated that the Matrix chat protocol can be simplified to the point where it becomes usable transparently with constrained devices.

2.9. Web browsing

Competing with HTTP [RFC9110] over its various transports; Gemini ([gemini]).

By virtue of cross proxying to HTTP, CoAP is generally capable of transporting web pages the same way as HTTP, albeit at a reduced feature set (in particular, most HTTP headers).

CoAP offers only niche benefits over HTTP when combined with HTML, the predominant markup language on the web: Any benefits of a more compact transport or implementation are dwarved by the typical size of pages and the complexity of the HTML ecosystem.

CoAP might be a suitable transport for Small Web environments such as Gemini [gemini], which can be rendered even by constrained devices.

2.10. E-Mail

While E-Mail was part of the considerations that led to the definition of the Proxy-Uri option (which would technically allow a

cross-proxy to accept POST requests to, say, mailto:office@example.com?subject=Sensor%20failure), no attempts are known to send or receive E-Mail over CoAP.

2.11. Video streaming

The use of CoAP for real time video streaming and telemetry from Unmanned Aerial Vehicles (UAVs) has been explored in [I-D.bhattacharyya-core-a-realist].

It is unclear whether CoAP could actually outperform unconstrained streaming protocols such as WebRTC, or whether devices that produce and consume video benefit from the constraints of CoAP.

3. References

3.1. Normative References

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

3.2. Informative References

[gemini] solderpunk, "Project Gemini, speculative specification, v0.16.1", 30 January 2022, <https://gemini.circumlunar.space/docs/specification.html>.

[CoJP] Vučinić, M., Ed., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", RFC 9031, DOI 10.17487/RFC9031, May 2021, <https://www.rfc-editor.org/rfc/rfc9031>.

[ace-key-groupcomm-oscore] Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-oscore-16, 6 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-oscore-16>.

[OpenThread_Multicast] Simon Lin, "Thread Border Router – Thread 1.2 Multicast", 15 March 2022, <https://openthread.io/codelabs/openthread-border-router-ipv6-multicast#0>.

[DHCP]
Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/rfc/rfc8415>.

[IGMP] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <https://www.rfc-editor.org/rfc/rfc3376>.

[I-D.ietf-core-coap-pubsub] Koster, M., Keränen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-coap-pubsub-11, 7 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-core-coap-pubsub-11>.

[I-D.ietf-core-comi] Veillette, M., Van der Stok, P., Pelov, A., Bierman, A., and I. Petrov, "CoAP Management Interface (CORECONF)", Work in Progress, Internet-Draft, draft-ietf-core-comi-11, 17 January 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-11>.

[RFC132] White, J., "Typographical Error in RFC 107", RFC 132, DOI 10.17487/RFC0132, April 1971, <https://www.rfc-editor.org/rfc/rfc132>.

[I-D.ietf-suit-manifest] Moran, B., Tschofenig, H., Birkholz, H., Zandberg, K., and O. Rønningstad, "A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest", Work in Progress, Internet-Draft, draft-ietf-suit-manifest-22, 27 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-suit-manifest-22>.

[I-D.draft-lenders-dns-over-coap] Lenders, M. S., Amsüss, C., Gündoğan, C., Schmidt, T. C., and M. Wählisch, "DNS over CoAP (DoC)", Work in Progress, Internet-Draft, draft-lenders-dns-over-coap-04, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-lenders-dns-over-coap-04>.

[RFC15] Carr, C., "Network subsystem for time sharing hosts", RFC 15, DOI 10.17487/RFC0015, September 1969, <https://www.rfc-editor.org/rfc/rfc15>.

[I-D.ietf-core-interfaces] Shelby, Z., Koster, M., Groves, C., Zhu, J., and B. Silverajan, "Reusable Interface Definitions for Constrained RESTful Environments", Work in Progress, Internet-Draft, draft-ietf-core-interfaces-14, 11 March 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-core-interfaces-14>.

[I-D.ietf-core-fasor]
Järvinen, I., Kojo, M., Raitahila, I., and Z. Cao, "Fast-Slow Retransmission Timeout and Congestion Control Algorithm for CoAP", Work in Progress, Internet-Draft, draft-ietf-core-fasor-01, 19 October 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-core-fasor-01>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[I-D.bhattacharyya-core-a-realist]
Bhattacharyya, A., Agrawal, S., Rath, H. K., Pal, A., and B. Purushothaman, "Adaptive RESTful Real-time Live Streaming for Things (A-REaLiST)", Work in Progress, Internet-Draft, draft-bhattacharyya-core-a-realist-02, 5 February 2019, <https://datatracker.ietf.org/doc/html/draft-bhattacharyya-core-a-realist-02>.

Appendix A. CoAP File Service

This sketches [TBD: describes] a file transfer protocol / remote file system built on top of CoAP.

A file server works similar to a WebDAV server, and follows these rules (which are sometimes expressed from the point of view of the server, but apply when a client maps them back into a file system in such a way that operations can round-trip):

* Directories are unconditionally represented by URIs with a trailing slash; files by those without one.

  The GET operation is used to list them (for there is no PROPFIND operation in CoAP). Different media types migth be used depending on the capabilities of the parties, with application/link-format as a base line.

  (Note that application/link-format is not particularly efficient for this purpose, as the directory listing needs to repeat the requested resource's full path for each entry).

  Clients need to be prepared for links that do not follow the regular pattern of a directory advertising its children, but (as with all unknown links) may ignore them.

* Paths are constructed by placing directory names and either an empty string (for the trailing slash) or the unescaped file name in Uri-Path options.

* Clients may attempt to treat any URI composed of the file server entry URI and additional path segments as files on the file server. Consequently, any additional services the file server may provide (e.g., as resources specified in extensions) are necessarily assigned URIs with a query, for these can not be inadvertedly constructed in an attempt to access a file.

* For lack of a HEAD option, file metadata can only be obtained by performing a GET on the directory containing the file, or a FETCH for efficiency if suitable media types are defined.

  All metadata is provided on a best-effort basis, and the supported directory formats limit what can be expressed. Typical supported metadata are the media type (expressed as ct in link format) and the size (sz).

* If write support is available and permissions allow, a client can create files by performing a PUT operation on a previously nonexistent resource.

  Files can be overwritten by PUTting a new representation. Files sent with block-wise transfer should be processed atomically by the server unless explicitly negotiated otherwise. (On POSIX file systems, this can be implemented without additional state by storing the blocks in a temporary file whose name contains the original file name and the block key of the request, and renaming it to the target name when receiving the last block).

  Directories can be created with PUT as well (it is clear from their path that they are directories); these would typically be empty and not have a content format, but may use a content format that already describes directory metadata. (It is up to the client to fall back to plain directory creation, if that is a viable option for it -- it would be if it's just about unimportant metadata, but might not if the initial metadata indicates an ACL).

* Files and directories can be removed using the DELETE operation, subject ot the same conditions as writing to resources.

  Deleting directories is recursive; client that desire POSIX semantics need to check whether the directory is empty and use the If-Match option with the empty directory's ETag to avoid race conditions.

* Regular CoAP extensions apply if the parties support them, for example:

  - Observation can be used to watch files or directories for changes.

- ETags (e.g. derived from the file's stats) can be used to ensure that large files are assembled correctly by the client, and to perform file updates with optimistic locking by using the If-Match option.

* Additional features can be specified and advertised separately, either per resoource or by a named specification that provides templates for further operations.

  For example, a specification might say that when a file system is advertised with a given interface (if parameter of link format), for each file and directory there is an associated resource at ?acl that describes access control applicable to that file, and can be used with GET and PUT as per the ACL's policies.

  Additional operations can use their custom media types and methods, and possibly create more resources. For example, a server-to-server copy (again, advertised by a suitable interface parameter) could provide a ?copy resource under any directory, to which a CBOR list containing two CRIs (source and destination) would be posted. That specification might describe that if copies are not completed instantly, the response might indicate a new location using Uri-Path and Uri-Query options (the latter might be necessary to not conflict with existing files) which tracks the status of the operation.

* File service is compatible with "index.html" style directory indices provided statically in the file system. It is recommended to not serve index files of typical directory listing formats (such as application/link-format) as these might mislead the client about the file system's content, and prevent clients that access the server as a file server from even seeing the providing file's name.

  These files still need to be written to or deleted in their file version (e.g. as "index.html"); the file server may use yet to be specified link relations to point out which files are being served as the index files.

Some implementation guidance should be provided around the interaction between idempotent requests that have no actual effect and preconditions: If a DELETE with If-Match is transmitted again on a new token (by a proxy relying on its idempotence), should the server respond Deleted rather than Not Found? (In this case, the client could at least know what to do with it, but) If a PUT with If-Match is transmitted again after it has been acted on, should the server respond Changed rather than Precondition Failed? (Probably "No" to both, as the former is easily recognizable by the client,

and the latter would delay faulting by long, but still needs further thought.)

Appendix B. Change log

From -02 to -03:

- *Added Runtime configuration section

- *Terminal access now has an implementation

From -01 to -02:

- *Added "Web browsing" section

From -00 to -01:

- *Added details to file service

Author's Address

Christian Amsüss
Austria

Email: christian@amsuess.com