“モノ”のインターネットへのつながり方 L4以上編

株式会社レピダム

前田 薫 (@mad_p)

Internet Week 2016 2016/11/30

lepidum
https://lepidum.co.jp/

自己紹介

- 名前
  - 前田 薫
- 所属
  - 株式会社レピダム
    シニアプログラマ
- コミュニティー活動
  - Lightweight Language
  - Identity Conference
  - http2勉強会

- 業務領域
  - 認証・認可、デジタルアイデンティティー、プライバシー
  - 標準化支援
  - ソフトウェアセキュリティー、脆弱性

はじめに

- L4 以上とは?
- 本パートで扱う内容
  - CoAP IoT向けHTTP (IETF WG core)
  - DTLS IoT向けのプロファイル (dice)
  - 関連技術 CBOR, CWTなど
  - 認証・認可 (ace)
- IETFで標準策定中の技術について紹介します

Constrained Device/Node

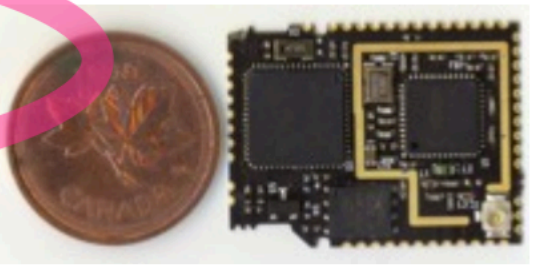
- IETFではIoTデバイスを「Constrained Device」と呼ぶことが多い
  - 直訳すると「制約されたデバイス」
  - CPU能力、メモリ、電源、ネットワークなどの資源に「制約」がある
- 本パートでは「制約デバイス」と書きます

Constrained nodes: orders of magnitude

10/100 vs. 50/250

- There is not just a single class of "constrained node"

- Class 0: too small to securely run on the Internet
  - "too constrained"
- Class 1: ~10 KiB data, ~100 KiB code
  - "quite constrained", "10/100"
- Class 2: ~50 KiB data, ~250 KiB code
  - "not so constrained", "50/250"



- These classes are not clear-cut, but may structure the discussion and help avoid talking at cross-purposes

http://www.ietf.org/proceedings/89/slides/slides-89-ace-2.pdf

COAP - THE CONSTRAINED APPLICATION PROTOCOL

CoAPの概観

- CoAPプロトコル (RFC 7252) coap.technology
  - リクエスト/レスポンスモデルでの制約デバイスとの通信に適したプロトコル
    - サーバー、クライアントの一方または両方が制約デバイス
  - HTTPをベースとする
    - URI、メディアタイプをサポート
    - HTTPとのマッピング
  - コンパクトなワイヤーフォーマット
  - 非同期通信 (UDP、再送制御)
  - ディスカバリー

IETF core WG

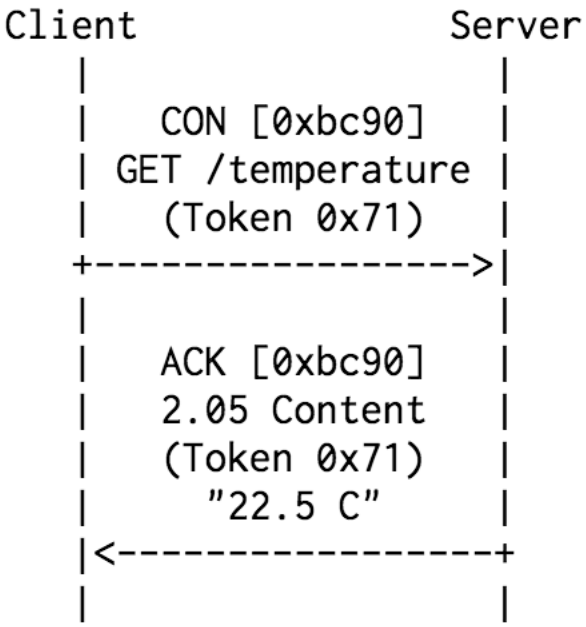
- core (Constrained RESTful Environments) 2010-
  - CoAPプロトコルに関する仕様策定を行う
- 主要なRFCおよびドラフト
  - RFC 6690 Link format
  - RFC 7252 CoAP
  - RFC 7390 Group Communication
  - RFC 7641 Observing
  - RFC 7959 Blockwise Transfers
  - draft-ietf-core-http-mapping HTTP mapping
  - draft-ietf-core-etch PATCH and FETCH methods
  - draft-ietf-core-resource-directory Resource Directory
  - draft-ietf-core-coap-tcp-tls over TCP, TLS, Websockets
  - draft-ietf-core-object-security Object Security

CoAPプロトコルとURI

- coap://s.example.com/light
- coaps://s.example.com/light
- デフォルトのポート番号
  - coap 5683
  - coaps 5684

リクエスト/レスポンス

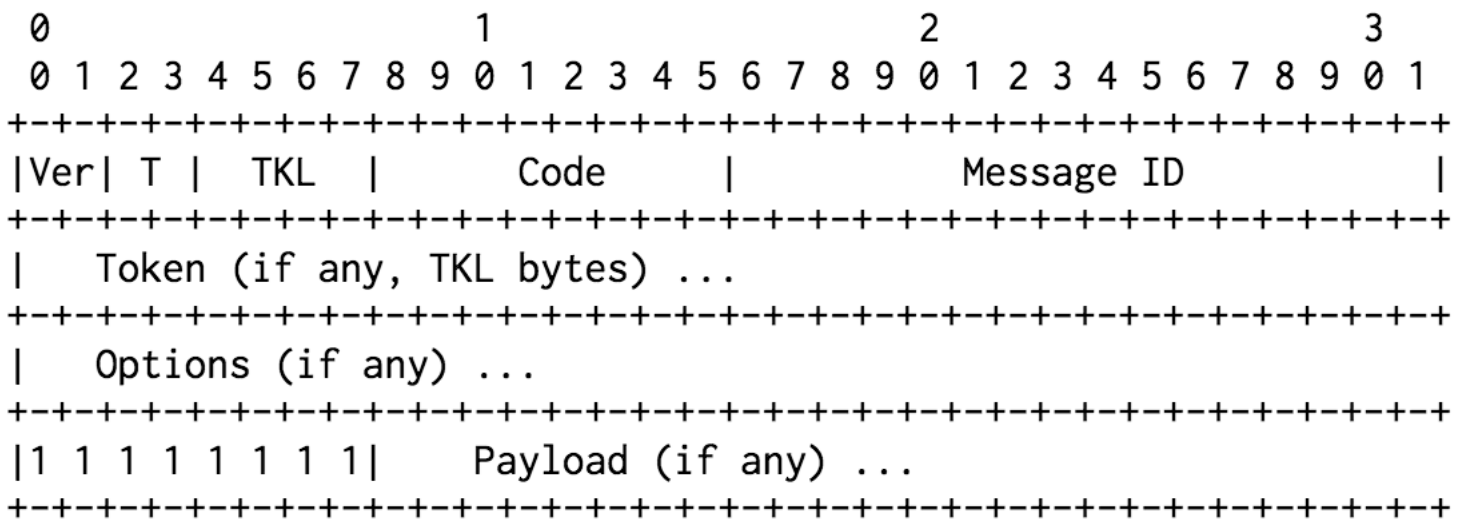
- /temperature をGETする例
  - CON/ACK, Tokenについては後述



バイナリフォーマット

- バイナリベースのコンパクトな通信
  - メッセージのパースがしやすい
  - code (メソッド、ステータス)は番号で指定
  - options (ヘッダ名に相当)は番号で指定
  - optionsの値もバイナリ
    - %エンコーディングやエスケープが不要
    - メディアタイプは番号で指定

CoAPワイヤーフォーマット



- Ver: バージョン 1固定
- T: タイプ、再送制御に使う
- Code: HTTPのメソッドとステータスコードに対応
- Message ID: 再送制御に使う
- Token, TKL: リクエストとレスポンスの対応に使う
- Options: HTTPのヘッダに相等
- Payload: HTTPのペイロードに相等

Code

- 8ビットフィールドを3+5に分ける
- リクエストではメソッドに対応
- レスポンスではステータスコードに対応

Code	Name	Reference
0.01	GET	[RFC7252]
0.02	POST	[RFC7252]
0.03	PUT	[RFC7252]
0.04	DELETE	[RFC7252]

Table 5: CoAP Method Codes

Code	Description
2.01	Created
2.02	Deleted
2.03	Valid
2.04	Changed
2.05	Content
4.00	Bad Request
4.01	Unauthorized
4.02	Bad Option
4.03	Forbidden
4.04	Not Found
4.05	Method Not Allowed
4.06	Not Acceptable
4.12	Precondition Failed
4.13	Request Entity Too Large
4.15	Unsupported Content-Format
5.00	Internal Server Error
5.01	Not Implemented
5.02	Bad Gateway
5.03	Service Unavailable
5.04	Gateway Timeout
5.05	Proxying Not Supported

Table 6: CoAP Response Codes

Options

- HTTPヘッダに対応
- Option番号と値の列
- 値の型: empty, uint, opaque, string
- 番号順に並べ、番号は差分でエンコード
- 4つのフラグ
- URIは分解される
  - Uri-*
- Content-Format
  - media-typeに相等

No.	C	U	N	R	Name	Format	Length	Default
1	x			x	If-Match	opaque	0-8	(none)
3	x	x	-		Uri-Host	string	1-255	(see below)
4				x	ETag	opaque	1-8	(none)
5	x				If-None-Match	empty	0	(none)
7	x	x	-		Uri-Port	uint	0-2	(see below)
8				x	Location-Path	string	0-255	(none)
11	x	x	-	x	Uri-Path	string	0-255	(none)
12					Content-Format	uint	0-2	(none)
14		x	-		Max-Age	uint	0-4	60
15	x	x	-	x	Uri-Query	string	0-255	(none)
17	x				Accept	uint	0-2	(none)
20				x	Location-Query	string	0-255	(none)
35	x	x	-		Proxy-Uri	string	1-1034	(none)
39	x	x	-		Proxy-Scheme	string	1-255	(none)
60			x		Size1	uint	0-4	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

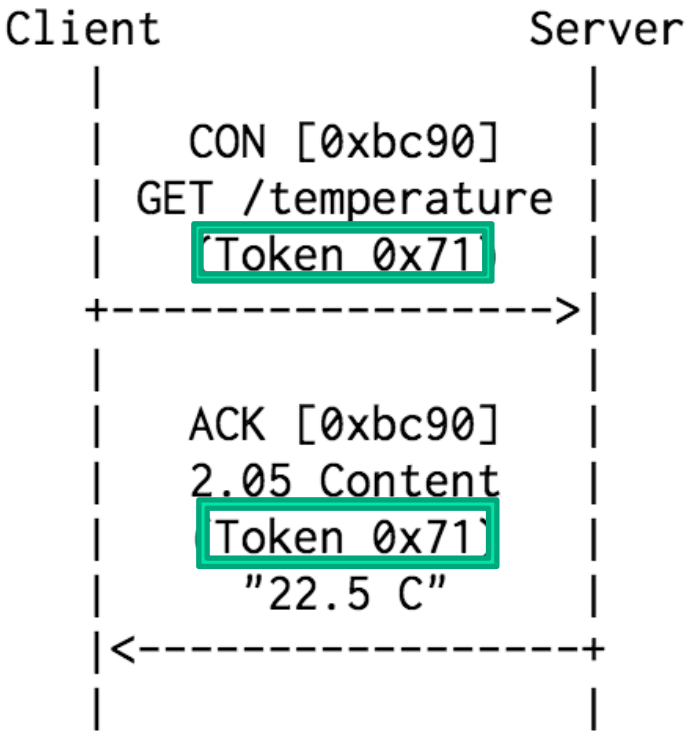
.

UDPベースの非同期通信

- 複数のリクエスト/レスポンスの平行通信
  - Tokenを使って、対応を管理
- 再送制御
  - 到達確認の必要性をタイプで示す
    - CON=必要、NON=不必要
    - ACK=到達確認、RESET
  - Message ID: 再送制御のための識別に使う
  - ACK piggy back
    - レスポンスをACKと一緒に送る
    - piggy back しないACKはCodeを0.00にする
  - タイムアウトとリトライ上限、exponential back-off

Token

- リクエストとレスポンスの対応を取る
- 非同期通信なので、複数のリクエスト/レスポンスが存在



再送制御の例

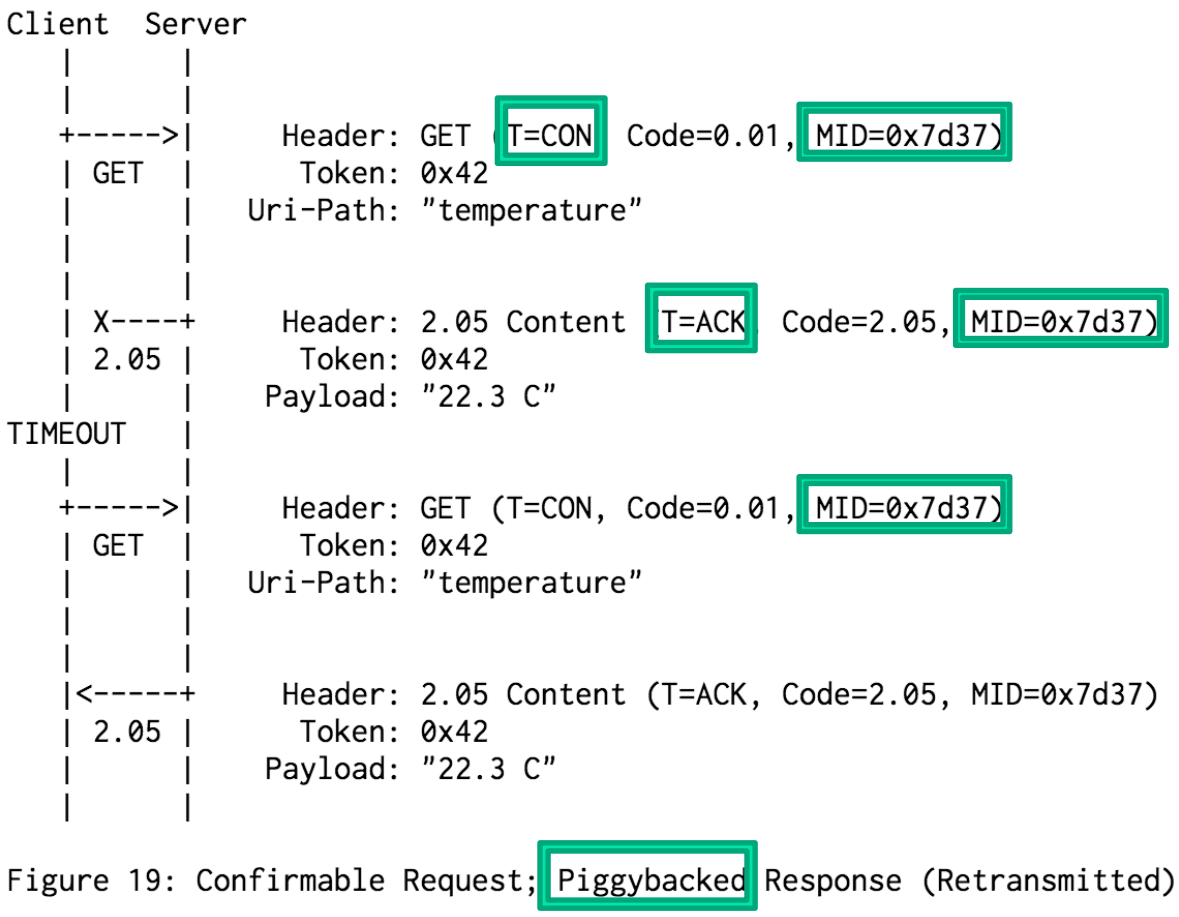


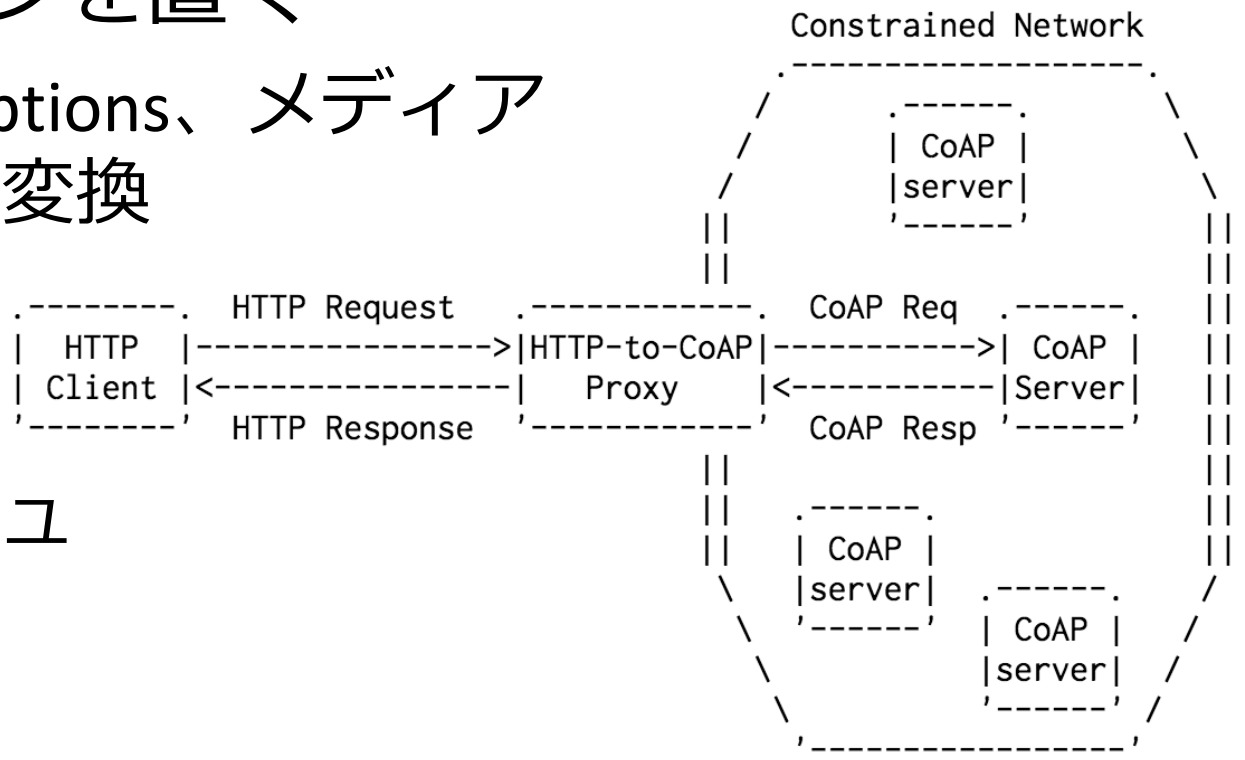
Figure 19: Confirmable Request; Piggybacked Response (Retransmitted)

CoAPとトランスポート

- CoAP over UDP/DTLS
  - UDPベースのCoAPをそのまま使う
  - セキュリティーの必要な場合はDTLS
  - IoT向けDTLSプロファイル
    - RFC7925 TLS/DTLS profiles for IoT
- CoAP over TCP/TLS
  - リライアブルなコネクションで使う場合
    - draft-ietf-core-coap-tcp-tls
  - ワイヤーフォーマットを変更
    - 再送制御の割愛
    - メッセージ長の指定
  - 設定情報、シグナリング(ping/pong, 接続終了など)
    - Code 7.xxを使用

CoAP-HTTPプロキシ

- draft-ietf-core-http-mapping
- HCプロキシを置く
  - Code、Options、メディアタイプの変換
  - キャッシュ



CoAPその他の仕様

- Discovery
  - RFC 6690 Link format
  - RFC 7252 CoAP
- Group Communication: RFC 7390
- Observing: RFC 7641
- Blockwise Transfers: RFC 7959
- draft-ietf-core-resource-directory
- draft-ietf-core-object-security

CoAPの実装

- http://coap.technology/impls.html

Constrained devices

Implementations for constrained devices are typically written in C.

Erbium

Contiki is a widely used operating system for constrained nodes, being employed for research and product development. Erbium is a full-fledged REST Engine and CoAP Implementation for Contiki.

View details »

libcoap

A C implementation of CoAP that can be used both on constrained devices (running operating systems such as Contiki or LWIP) and on a larger POSIX system. Moreover, the library has

CoAP is not only used between constrained devices, but also between them and more powerful systems such as cloud servers, home centrals, smartphones:

Server-side

Java

One significant Java-based implementation of CoAP is **Californium**.

View details »

nCoAP is a Java implementation of the CoAP protocol using the Netty NIO client server framework:

View details »

Leshan is an OMA Lightweight M2M (LWM2M) server-side implementation, on top of Californium.

View details »

Browser-based

Copper is an extension for Firefox to enable direct access to CoAP resources from a browser.

View details »

Smartphones

Some implementations are specifically targeting mobile devices such as smartphones and tablets. These tend to differ between platforms:

iOS, OSX

A simple iOS client implementation has been written by Wojtek Kordylewski in Objective-C.

View details »

CoAP client and server libraries are also available in Swift:

View details »

Android

CoAPまとめ

- UDPベース
  - TCPハンドシェイク不要、リクエスト/レスポンスの並行通信、確認不要メッセージ
- バイナリフォーマット
  - パースしやすい
- 応用に合わせた関連仕様

TLS/DTLS IOT PROFILES

IETF dice WG

- dice (DTLS In Constrained Environments) 2013-2016
- IoT向けDTLS profile
  - CoAP通信のセキュリティーはDTLSを使用
  - IoTの特性を反映したプロファイル
  - 認証/共通鍵/公開鍵/証明書ハンドリングの解説
  - 暗号スイート/キー長/証明書/などの推奨設定
- RFC7925 TLS/DTLS profiles for IoT

COAP RELATED STANDARDS

CoAP関連技術

- CBOR cbor.io
  - RFC7049
  - コンパクトなバイナリフォーマット
    - media-type: application/cbor
  - MessagePackのIETF版
- CWT (コット)
  - draft-ietf-ace-cbor-web-token
  - 署名・暗号化可能なCBOR
  - JWT (ジョット)のJSONをCBORにした版

AUTHENTICATION AND AUTHORIZATION FOR CONSTRAINED ENVIRONMENTS (ACE)

制約デバイスにおける認証・認可

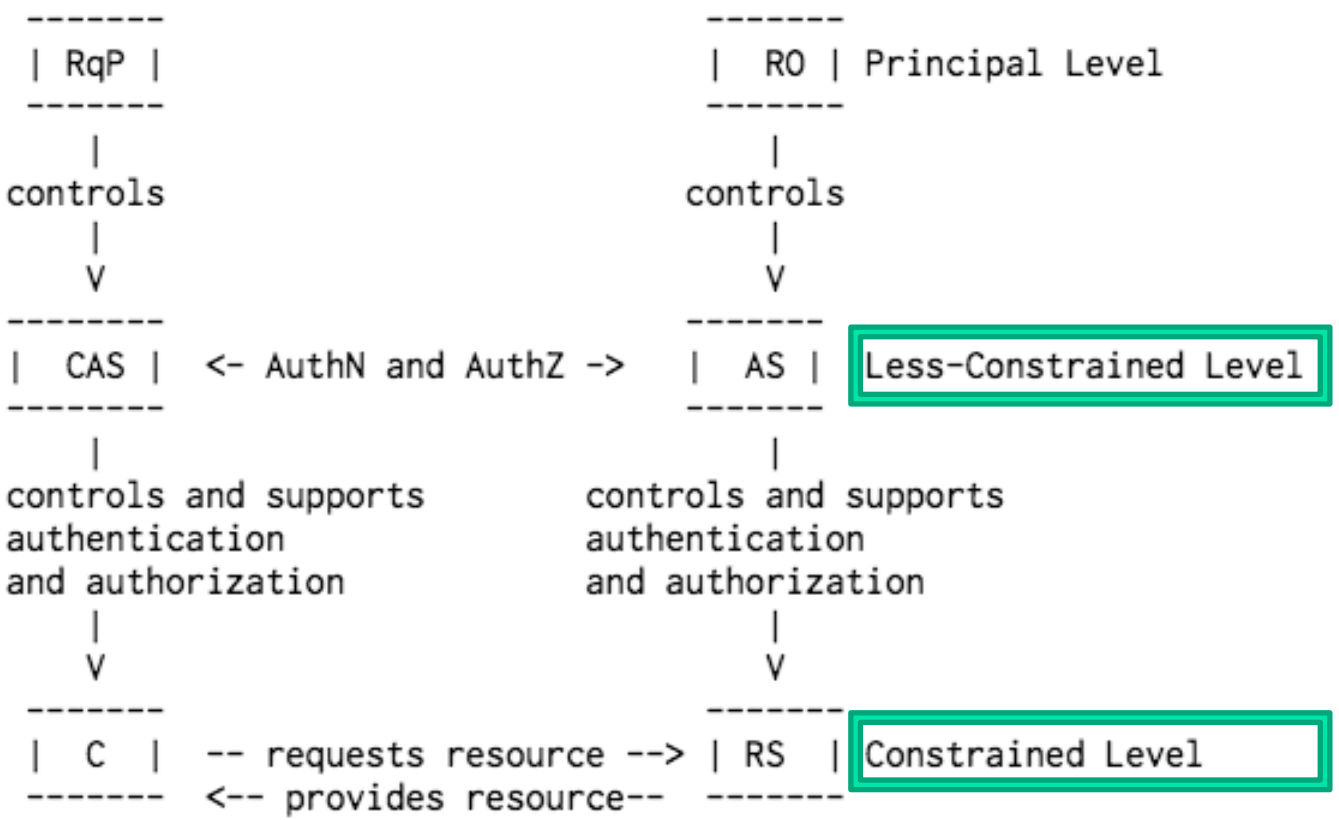


Figure 3: Overall architecture

IETF ace WG

- ace (Authentication and Authorization for Constrained Environments) 2014-
  - 制限された環境での認証・認可を検討
  - IETFではIoT文脈における認証・認可技術全般も「ace」と呼ぶことがある
  - まだユースケースがRFCになったばかり
- 主要なRFCおよびドラフト
  - RFC 7744: usecases
  - draft-ietf-ace-actors actors
  - draft-ietf-ace-cbor-web-token CWT
  - draft-ietf-ace-oauth-authz oauthによるソリューション

IoTにおける認証・認可検討の流れ

- ユースケースを集める
- 登場人物のモデルを明らかにする
- 既存技術(OAuth)の適用を考える

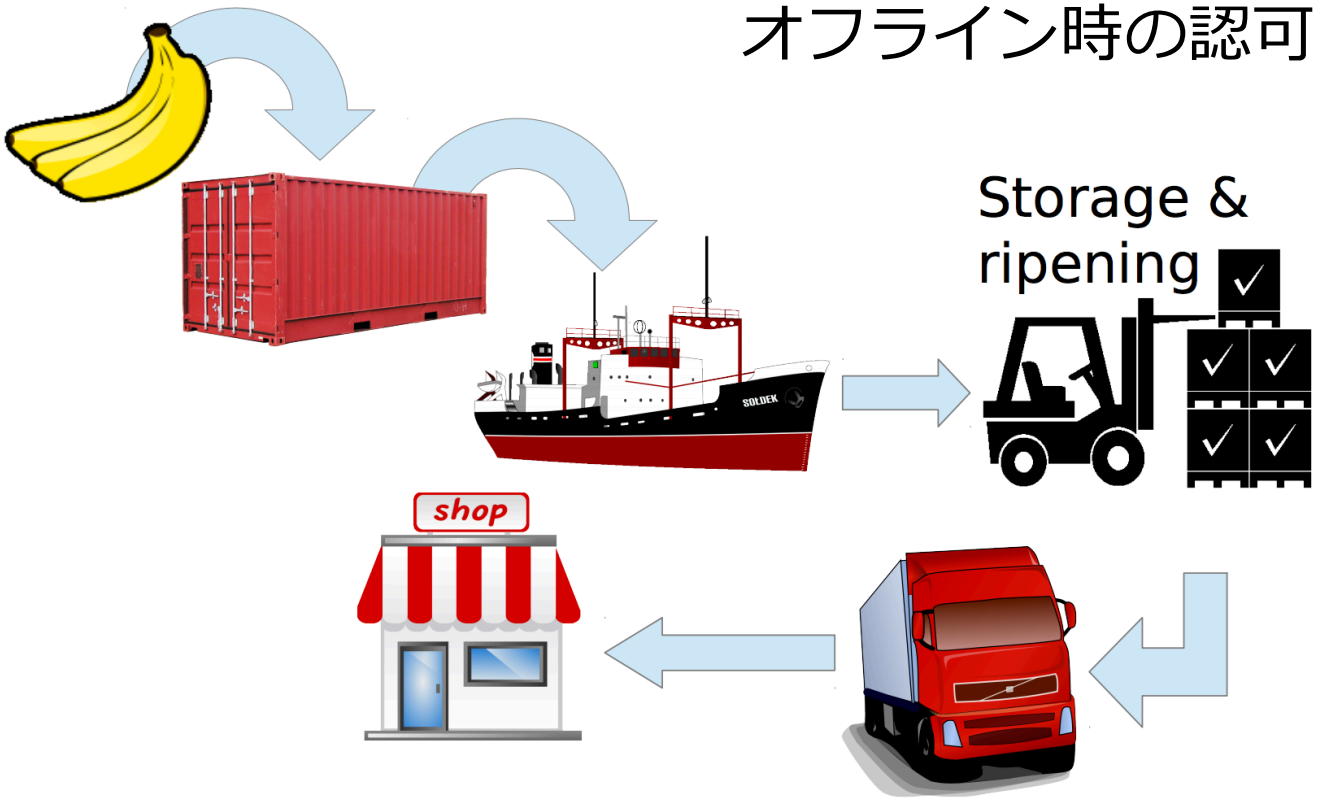
- 今日の話はドラフトを元にします
  - RFCまではもう少し時間がかかりそう

RFC 7744: Use Cases

- 6つのユースケースが提示されている
  - コンテナモニタリング
  - ホームオートメーション
  - パーソナルヘルスケアモニタリング
  - ビルディングオートメーション
  - スマートメーター
  - スポーツ&エンタテインメント
  - 産業制御システム

ace Use Cases

Container Monitoring



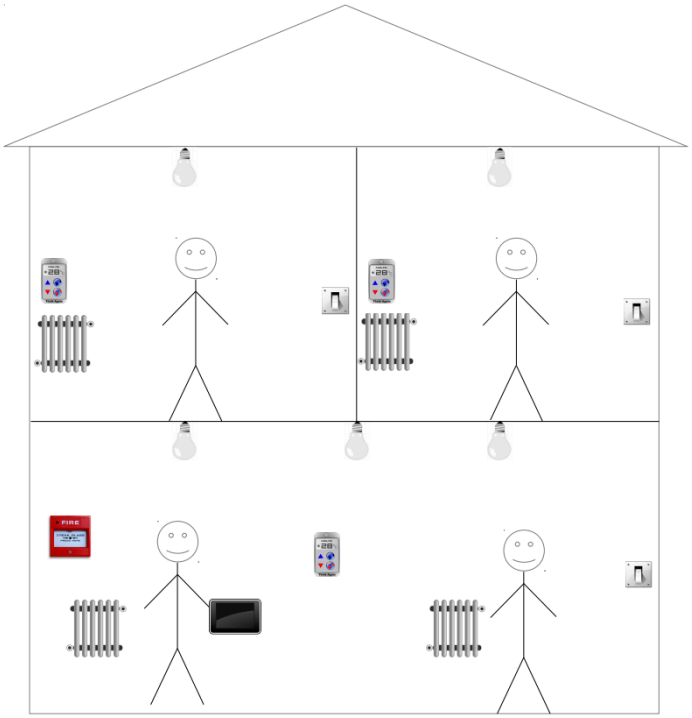
ace Use Cases

Home Automation

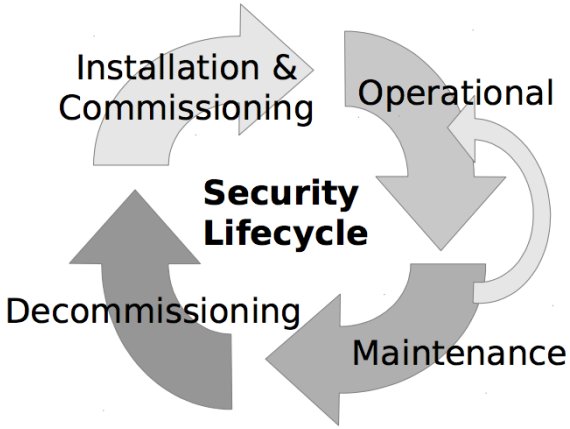
アクセス権のリモート委譲

ace Use Cases

Building Automation

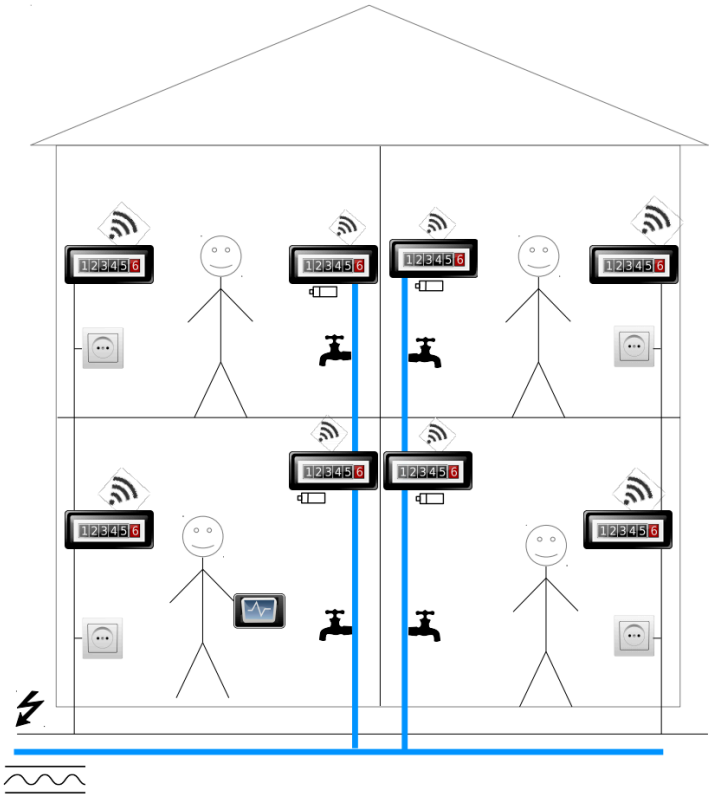


セキュリティー
ライフサイクル



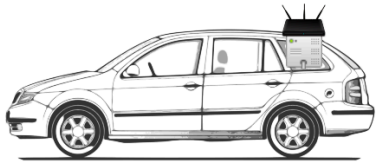
ace Use Cases

Smart Metering



電力センサーへの
水・ガスの相乗り

Base station



Actors:登場人物モデル

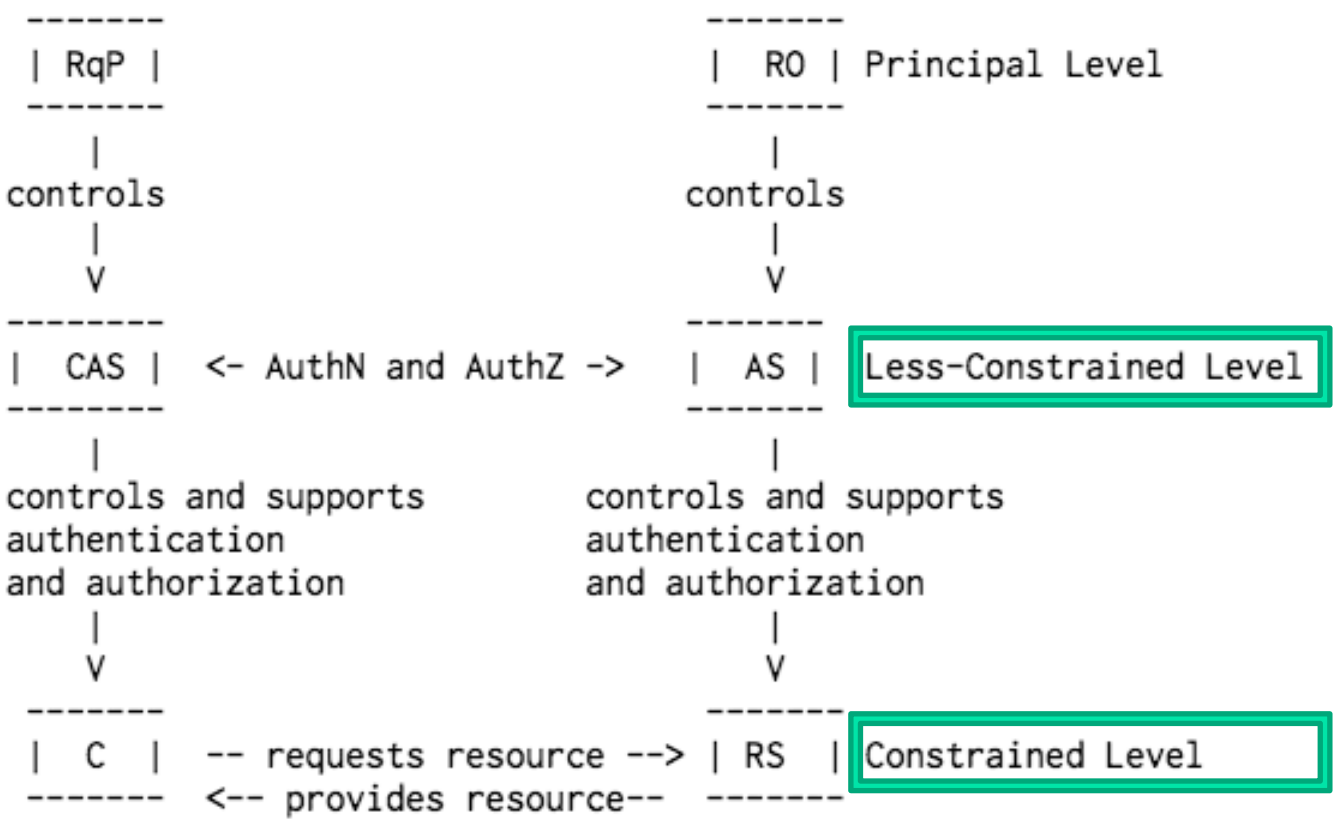
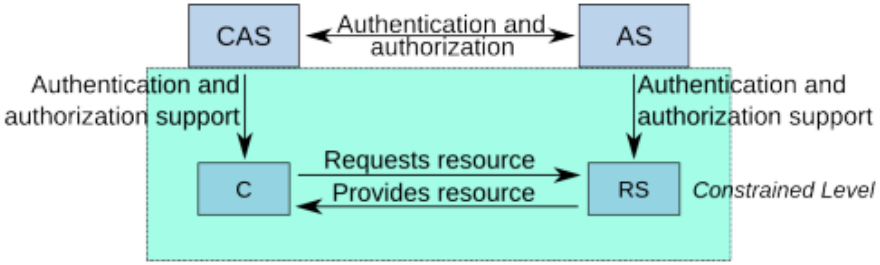


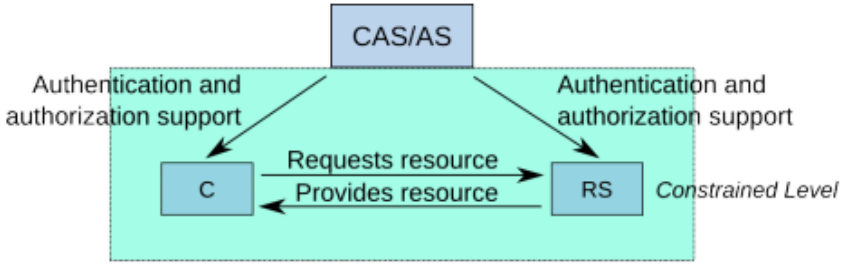
Figure 3: Overall architecture

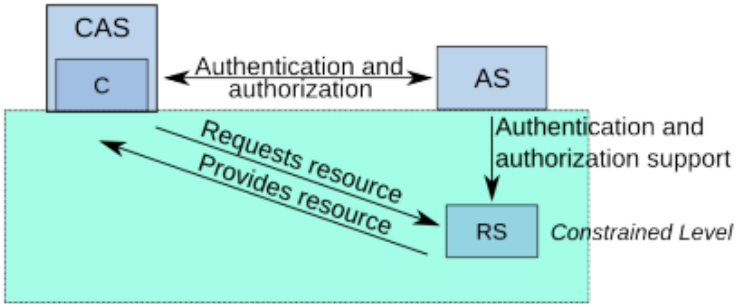
Less-Constrained Level

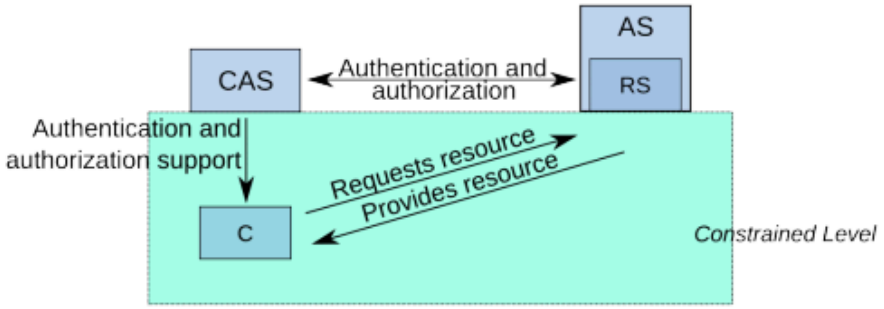
- 制約されていないデバイス層を考える
- 認証・認可の複雑な処理はless-constrained levelで処理
  - 署名つきトークンなどで制約デバイスに渡す
  - 制約デバイスでも署名検証程度の暗号計算はできるという前提

ace Authorization Model









https://www.ietf.org/proceedings/94/slides/slides-94-ace-2.pdf

OAuth Profile for IoT

- draft-ietf-ace-oauth-authz
- OAuth 2.0のToken Endpoint, Introspect Endpointを拡張して、IoT向けとする仕様
- JSON, JWT, x-www-form-urlencodedをCBOR, CWTで置きかえる
  - 頻出のキーワード、文字列を数値にマップ
- トランスポートとして、CoAP/DTLSを仮定
- bearerトークンでなくpop (proof-of-possession)トークンを標準とする
- less-constrainedレベルへのトークン検証問合せなど

Aceまとめ

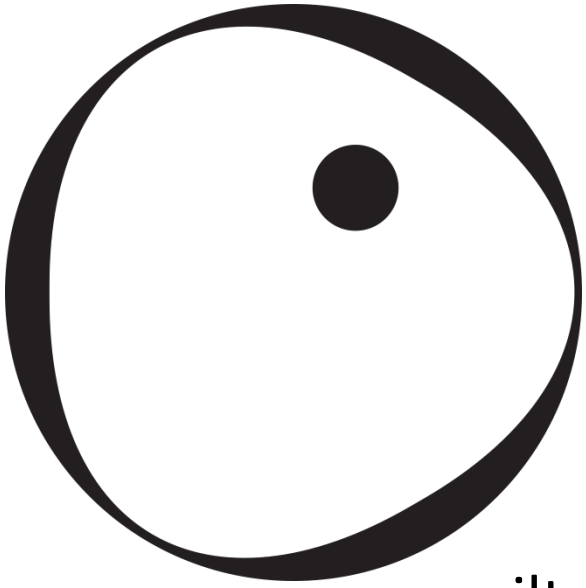
- IoT文脈における認証・認可技術を検討
- ユースケースからの要求抽出
- less-constrainedレベルによる認可処理
- OAuthプロトコルのIoT向けプロファイル
  - CBOR/CWTをベースとしたパラメータ、トークンフォーマット

SUMMARY

まとめ

- CoAP
  - リクエスト/レスポンスベースのIoT向け通信プロトコル
- CoAP/DTLS
  - IoT機器向けのDTLSプロファイル
- CBOR, CWT
  - バイナリーデータフォーマット、署名、暗号化
- ACE
  - 制約環境での認証・認可
  - less-constrained levelの活用

Any Questions? / Feedbacks Welcome!



lepidum

https://lepidum.co.jp/

mailto:maeda@lepidum.co.jp / twitter: @mad_p