Implementation of M2M applications for demonstrating benefits of ICN architecture

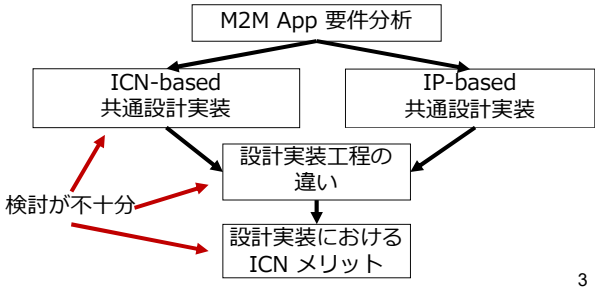
大阪大学大学院情報科学研究科
情報ネットワーク学専攻
村田研究室　高 宇豪

Information-centric networking (ICN)

- 効率性とロバスト性などにおける問題を総合的に解決するネットワーク・アーキテクチャ
  - IoT シナリオで分散型処理に親和性が高い
  - Primitive name resolution: 名前解決は NW (ICN) 層でプリミティブに行うので、App 開発者はサービスのロケーションを知る必要がなく、サービス ID のみで通信可能
  - Programmable: ルータ (NW 層) は任意処理を実行可能で、それにより分散処理を効率化可能で、より多くの計算リソースを活用可能

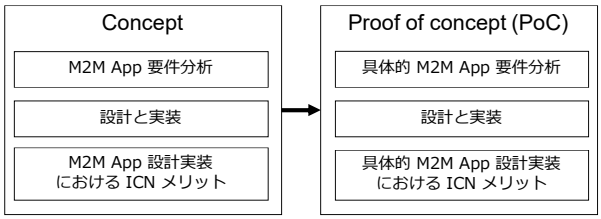
ICN の課題

- M2M App の ICN-based 共通設計実装方法および設計実装工程における ICN メリットの検討が不十分
  - よりバリエーション多くの App を実装するために共通の設計実装方法が必要



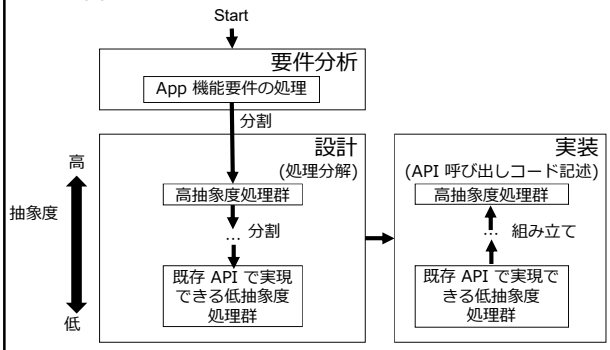
研究目的とアプローチ

- 目的：M2M App の ICN-based 設計実装方法および設計実装工程における ICN メリットを示す



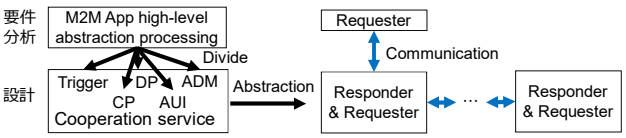
App 実現方法

- App は要件分析と設計と実装の工程で実現可能



M2M App 要件分析と設計

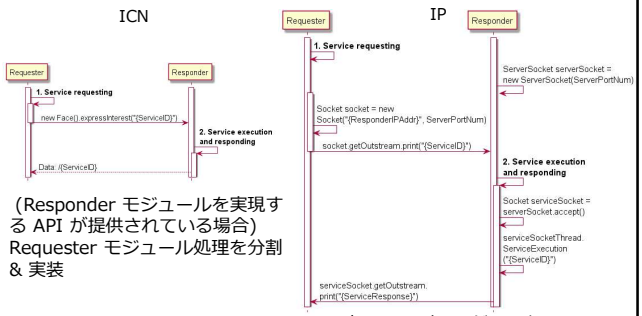
- 機能要件の処理は M2M 標準サポートユースケース [1] における以下ノード役割と処理に分割可能
  - Trigger: サービスのトリガ
  - Application decision-maker (ADM): サービス連携
  - Application user interface (AUI): 自身のソフトウエアとハードウエア状態の更新
  - Content provider (CP): コンテンツデータ提供
  - Data processor (DP): データプロセシング・サービス提供
  - プログラム構造の理解が簡単なために、App 処理はサービス要求者 Requester とサービス応答者 Responder 間通信の連鎖に抽象化



[1] oneM2M, "Use Case Collection", Accessed: 2019-02-08, http://www.onem2m.org/images/files/deliverables/Release2/TR-0001-Use_Cases_Collection-V2.4.1.pdf

M2M App 設計と実装

- Requester-Responder 間通信はサービス要求転送、サービス実行と応答転送に分割可能



（Responder モジュールを実現する API が提供されている場合）
Requester モジュール処理を分割 & 実装

Requester と Responder モジュール処理を分割 & 実装

M2M App 設計と実装工程の違い

	ICN	IP
設計	• NW 層 API レベルまで分割 ➢ 計算リソースを利用するサービスは NW 層計算リソースを利用すべきで、そのサービスを NW 層 Web API として利用すべきで分解する必要がない ➢ 他の App との共通サービスは再利用のために、 ICN フレームワーク API と利用すべきで分解する必要がない	• スクラッチ用 API レベルまで分割 ➢ フレームワークか Web API は必ずしも発見/呼び出し/実行できないため、それらの API 組み立てによる実装はできない ➢ App サービスをスクラッチで実装
実装	• Web API かフレームワーク API の組み立て • Web API 呼び出しはサービス ID が必要	• スクラッチ • Web API 呼び出しはサービス ID とサービス・ロケーションが必要

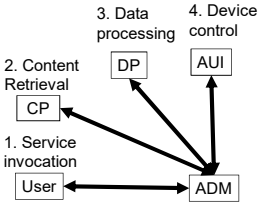
IP と比較した ICN メリットとデメリット

メリット	デメリット
• App 開発者は App 層コードを書く (実装) 量が少ない ➢ ICN は NW 層 API の組み立てで、IP はスクラッチで実装するため • API 呼び出しはサービス・ロケーションを考慮せずに、サービス ID のみで実現可能	• App 開発者は未学習フレームワークか Web API 学習コストが発生 ➢ ただし、多くの場合では、開発者は API 知識が豊富なドメインの App を開発するため、未学習 API の数は少なく学習コストは小さい

具体的 App 要件分析と設計

- 道案内システムで案内ドローンが移動と待機を繰り返すように移動経路を計算してドローン制御
  1. Service invocation: 道案内ノード連携サービスの呼び出し
  2. Content retrieval: 移動経路計算のためのデータの取得
  3. Data processing: 移動経路計算
  4. Device control: ドローンデバイス制御の既存 API の呼び出しによるデバイス制御

Operation	Service ID (API name)
1. Service invocation	/Guidance/{UserID}/{TargetID}
2. Content retrieval	/{ContentID}
3. Data processing	-
4. Device control	/{DeviceID}/{Instruction}



具体的 App 実現可能性の検証実験

- 下記シナリオの App 機能を実現したことを確認し、ICN-based 実装設計方法は道案内 App の機能を実現できることを示した
  - ➢シナリオ：(0, 0) → (10, 0) → (20, 0) → (20, 10) → (20, 20) の経路に沿ってドローンの移動と待機
  - ➢(x, y) = (原点から北方向距離、原点から東方向距離)

1. Service invocation
[User] Sent packet: Guidance User0 Target0
[User] Received packet: ACK
[DCS] Received packet: Guidance User0 Target0
[DCS] Sent packet: ACK
2. Content retrieval and 3. Data processing
[CP] Received packet: DeviceInfo Area0
[CP] Sent DeviceInfo: FR0
[DCS] Received DeviceInfo: FR0
[DCS] Send Device Control Request: FR0 FNPath 0.0,0.0 10.0,0.0 20.0,0.0 20.0,10.0 20.0,20.0
4. Device control
[GD] Received packet: FR0 FNPath 0.0,0.0 10.0,0.0 20.0,0.0 20.0,10.0 20.0,20.0
[GD] Sent the Device Control response: ACK

4. Device control
Now Location: (10.00322558, 0.4122134499) ← 移動
The remains of waypoints: (3)
Remained HoverTime:4.0 ← 待機
Now Location: (20.01231418, 0.1451367656)
The remains of waypoints: (2)
Remained HoverTime:5.0
Now Location: (20.02325234, 9.4599345392)
The remains of waypoints: (1)
Remained HoverTime:4.0
Now Location: (20.00322558, 15.4029904499)
The remains of waypoints: (0)
Remained HoverTime:4.0

設計実装工程における ICN メリット

- ICN では App 開発者の実装量は IP の 14.2%
  - ➢ 理由としては、ICN はフレームワークか Web API 組み立て方式で、IP はスクラッチで実装するため
  - ➢ 経路計算モジュール DP は NW 層計算リソースを使用し、ノード連携とコンテンツ提供とデバイス制御モジュール ADM, CP, AUI は他の App と共用できるため、それらのモジュールは NW 層 API として使用

ICN modules (lines)	IP modules (lines)
User 37	User 28
	ADM 63
	CP 34
	DP 74
	AUI 56
Total 37	Total 261
Ratio: 14.2%	

コンテンツ提供モジュール CP 実装例

- ICN では App 層コードがなく、IP ではスクラッチ
  - ➢コードが実現する処理はモジュール配置と設定後の機能要件処理

IP-based Code

ICN ではこのモジュールを App 層で実装する必要がない

```
public static void main(String[] args) {

  try {
    ServerSocket ConnectionWaitingSocket = new ServerSocket(Common.CPPort);
    Socket NewConnectionSocket = ConnectionWaitingSocket.accept();
    BufferedReader reader_NewConnectionSocket = new BufferedReader(
        new InputStreamReader(NewConnectionSocket.getInputStream()));
    PrintWriter writer_NewConnectionSocket = new PrintWriter(
        new BufferedWriter(new
        OutputStreamWriter(NewConnectionSocket.getOutputStream())));

    String str;
    while (true) {
      if ((str = reader_NewConnectionSocket.readLine()) != null) {
        System.out.println("[CP] Received packet: " + str);

        if (str.contains("DeviceInfo Area0")) {
          writer_NewConnectionSocket.println("FR0");
          writer_NewConnectionSocket.flush();
          System.out.println("[CP] Sent DeviceInfo: FR0");
        }
      }
      Thread.sleep(1000);
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

ソケットとストリーム生成

パケット待ち受け

コンテンツ転送

設計実装工程における ICN メリット

- ICN では Web API 呼び出しはサービス・ロケーションを考慮せずに、サービス ID のみで実現可能

ICN

```
try {
  Face GuidanceRequestFace = new Face();
  GuidanceRequestFace.expressInterest(new Name("/Guidance/User0/Target0"));
} catch (Exception e) {e.printStackTrace();}
```

サービス ID

IP

```
try {
  socket = new Socket("localhost", Common.DCS_RecepitonPort);
  writer = new PrintWriter(socket.getOutputStream(), true);
  writer.println("Guidance User0 Target0");
} catch (Exception e) {e.printStackTrace();}
```

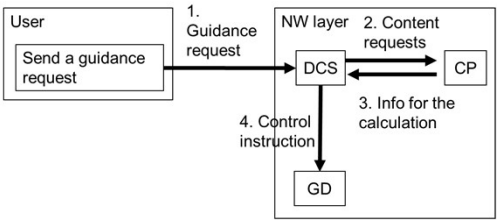
サービスロケーション

サービス ID

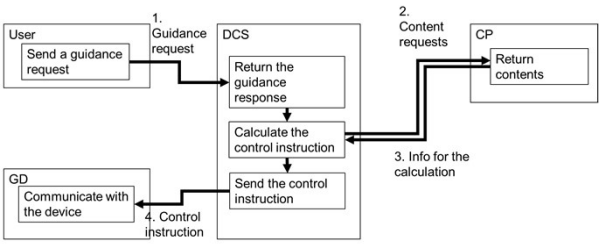
まとめと今後の課題

- まとめ
  - ➢ICN-based M2M App 共通の設計実装方法とメリットの検討が不十分
  - ➢ICN は API 組み立て方式で、IP はスクラッチ方式で実装
  - ➢本研究提示の設計実装方法は具体的 App を実現できることを示した
  - ➢ICN の App コード実装量が IP の 14.2 %であることを示した
- 今後の課題
  - ➢ICN-based App 開発方法 (要件分析、設計、実装、テスト、配置、保守) の提案と作業量の評価

付録 ICN-based PoC プログラムの設計



付録 IP-based PoC プログラムの設計



付録 実験で用いたソフトウエア

Software	Description
jndn	The Java NDN library
NFD 0.5.0	The NDN forwarding engine
DroneKit-Python 2.0	The simulator of the drone mobility
Mission Planner 1.3.44	The GUI tool monitoring the state of drones
Java 8	The language used in the IP and the ICN program
Ubuntu 14.04 LTS	The OS executing the program