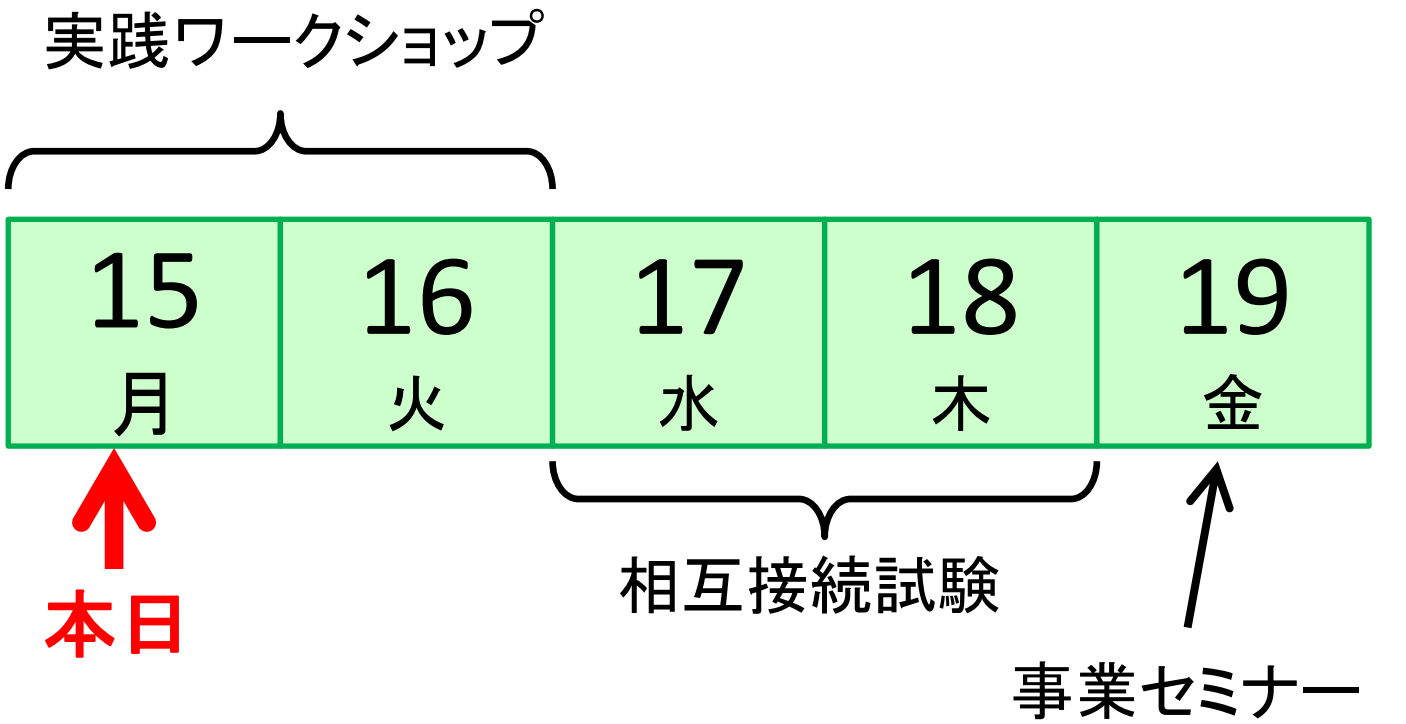
IEEE1888実践ワークショップ
～１日目～

主催: 東大グリーンICTプロジェクト
　　プロトコル標準化WG
場所: 東京大学 工学部2号館 電気系会議室５
日程: 2012年10月15日-16日

はじめに

- 今週(15日-19日)は IEEE1888 Week です。



http://www.ssk21.co.jp/seminar/S_12347.html

はじめに

- 実践ワークショップ開催の目的
  - IEEE1888 に関連する機器開発、ソフトウェア開発、サービス開発のノウハウを、一堂に会して共有し、各社での採用検討・製品化への足掛かりとする。

- カリキュラム
  - 第1日目
    - IEEE1888の背景・技術・実現事例について理解する
    - SDK(ソフトウェア開発キット) を使って IEEE1888プログラミングを理解する
  - 第2日目
    - 大規模 M2M (Machine-to-Machine) システム の実現について理解する
    - IEEE1888システムの設計・構築について理解する

本日(15日)のスケジュール

10:00　IEEE1888とは何か？（全体像）

11:00　IEEE1888機器を使ってみる

12:00　昼食

13:00　IEEE1888ソフトウェア開発キット(SDK)の概要

13:30　SDKを使ったFETCH手順のプログラミング

15:00　休憩

15:30　SDKを使ったWRITE手順のプログラミング

16:30　C言語版/Arduino版についての紹介

17:00　解散

明日(16日)のスケジュール

10:00　IEEE1888レジストリの仕組み

10:30　レジストリを用いたCEMS実現への取組み

11:00　IEEE1888レジストリとの通信プログラミング実習

12:00　昼食

13:00　IEEE1888システムの設計と構築: その1

14:30　休憩

15:00　IEEE1888システムの設計と構築: その2

16:00　簡易な相互接続実験

17:00　解散

本日(15日)のスケジュール

10:00　IEEE1888とは何か？（全体像）

11:00　IEEE1888機器を使ってみる

12:00　昼食

13:00　IEEE1888ソフトウェア開発キット(SDK)の概要

13:30　SDKを使ったFETCH手順のプログラミング

15:00　休憩

15:30　SDKを使ったWRITE手順のプログラミング

16:30　C言語版/Arduino版についての紹介

17:00　解散

IEEE1888 とは何か？

IEEE(米国電気電子学会)が 2011年に策定した
国際的な通信規格の一つ

「次世代の監視制御のための通信規格」
として開発された

Ubiquitous Green Community Control Network (UGCCNet)
と呼び、
1．コミュニティーの電力管理・設備管理
2．ITシステムとの連携
などを得意とする

(*) 日本では、FIAP (Facility Information Access Protocol) と呼ぶこともある

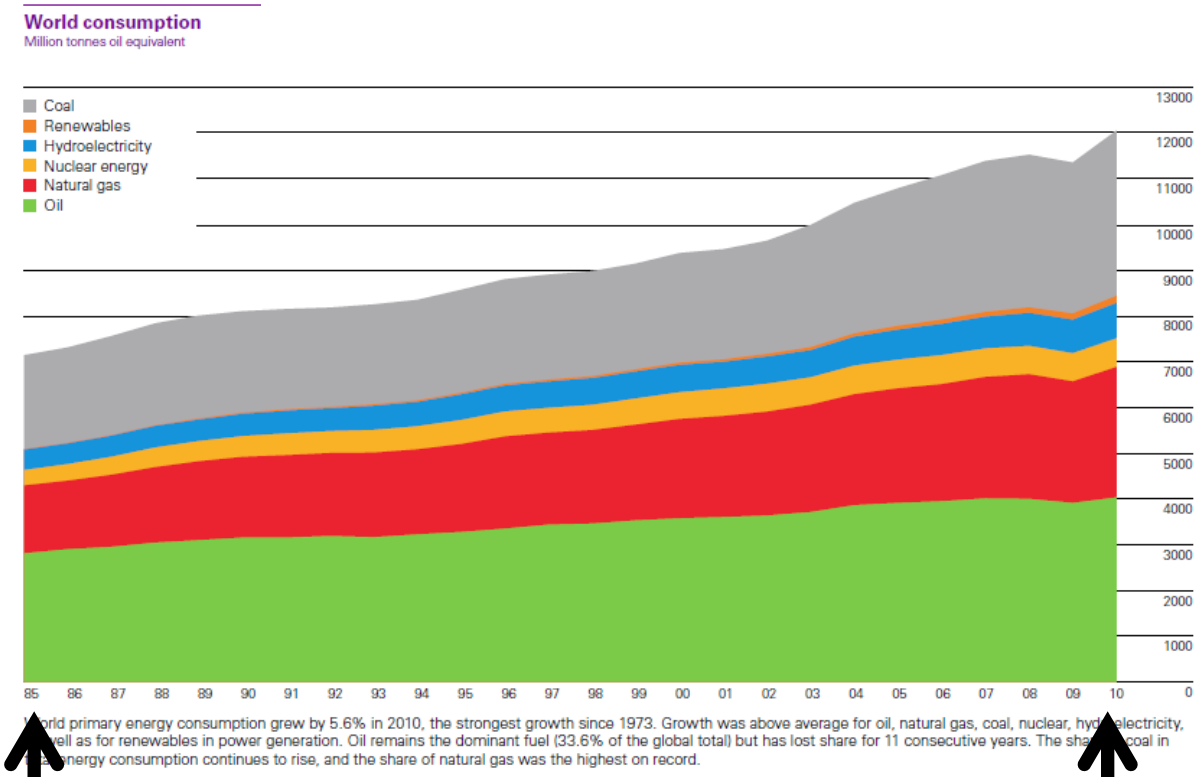
IEEE1888とは何か？

- **背景、目的、概要**
- アーキテクチャ
  - コンポーネント (GW, Storage, APP)
  - 通信手順 (WRITE, FETCH, TRAP)
  - システムの実現方法
- データ構造
  - メッセージ・フォーマット
  - “ポイント”の概念
  - 具体例
- 参照機器モデル
- 東大の5万kW電力需要家の管理事例

IEEE1888 を取り巻く時代背景：その１

厳しくなるエネルギー問題

世界のエネルギー消費量
(過去25年間でおよそ倍に)



1985

引用: BP Statistical Review of World Energy

2010

* 懸念
- 埋蔵資源の限界
- 地球温暖化

* 重要なミッション
- エネルギーの管理
  (e.g., 特に電力)
- 再生可能エネルギーの利用
  (e.g., 風力、太陽光)

* 技術
情報通信基盤(ICT)
- 自動最適制御
- 大容量の管理能力

IEEE1888 を取り巻く時代背景：その２
コンピュータ・システムの進化

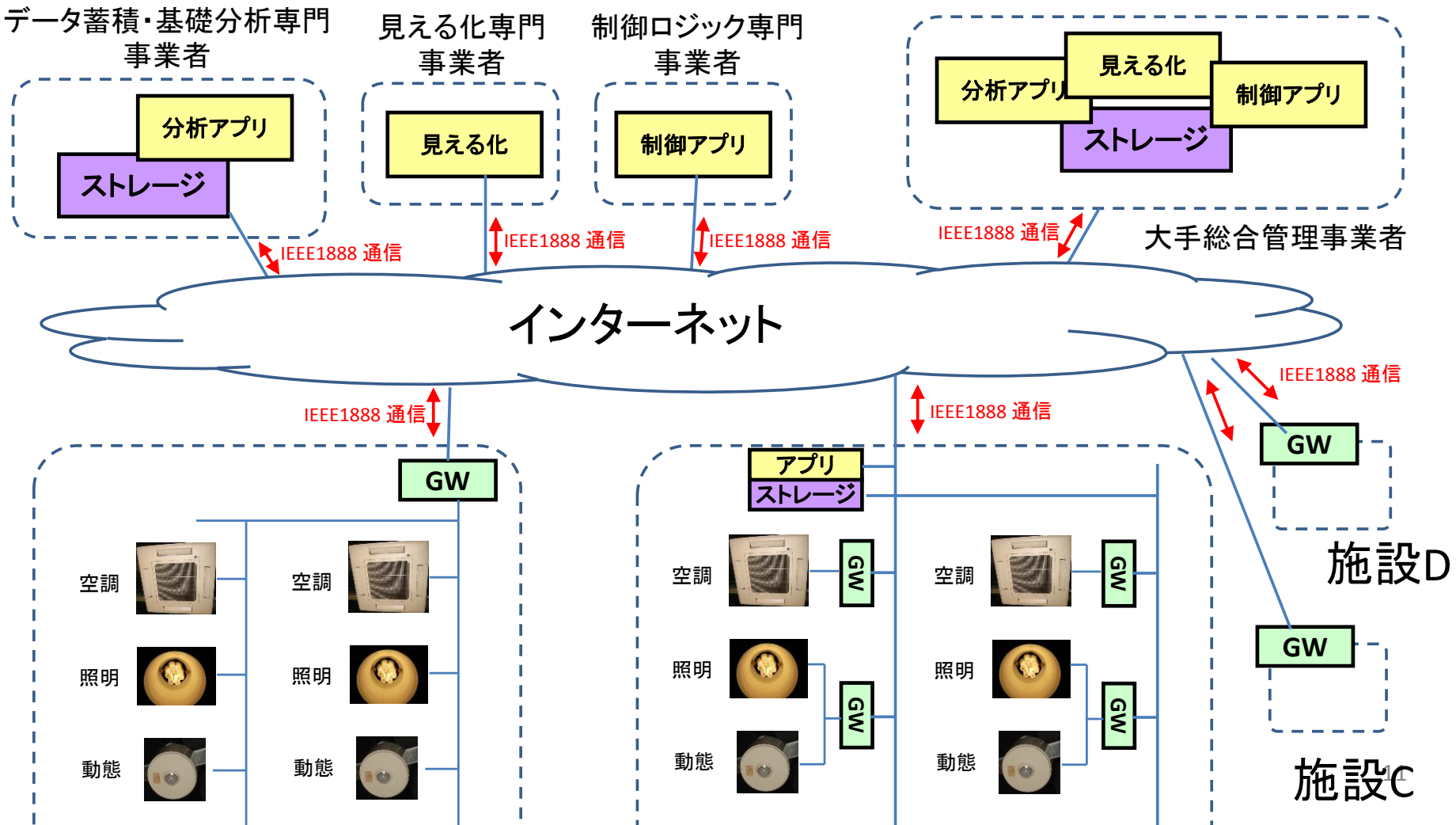
- 設備ネットワークが誕生した時代 (1990年代～)
  - 物理バスに計測機器や設備機器を接続
    - 標準バスネットワーク (RS485など)
    - 標準アプリケーション・プロトコル (Modbus, BACnet など)
    - 一体標準型 (Lonworks)
  - 集中管理（帳票出力など）は、デスクトップ・パソコンで。
    - 専用アプリをインストールして使用

当時は、最も現実的で最良の方法

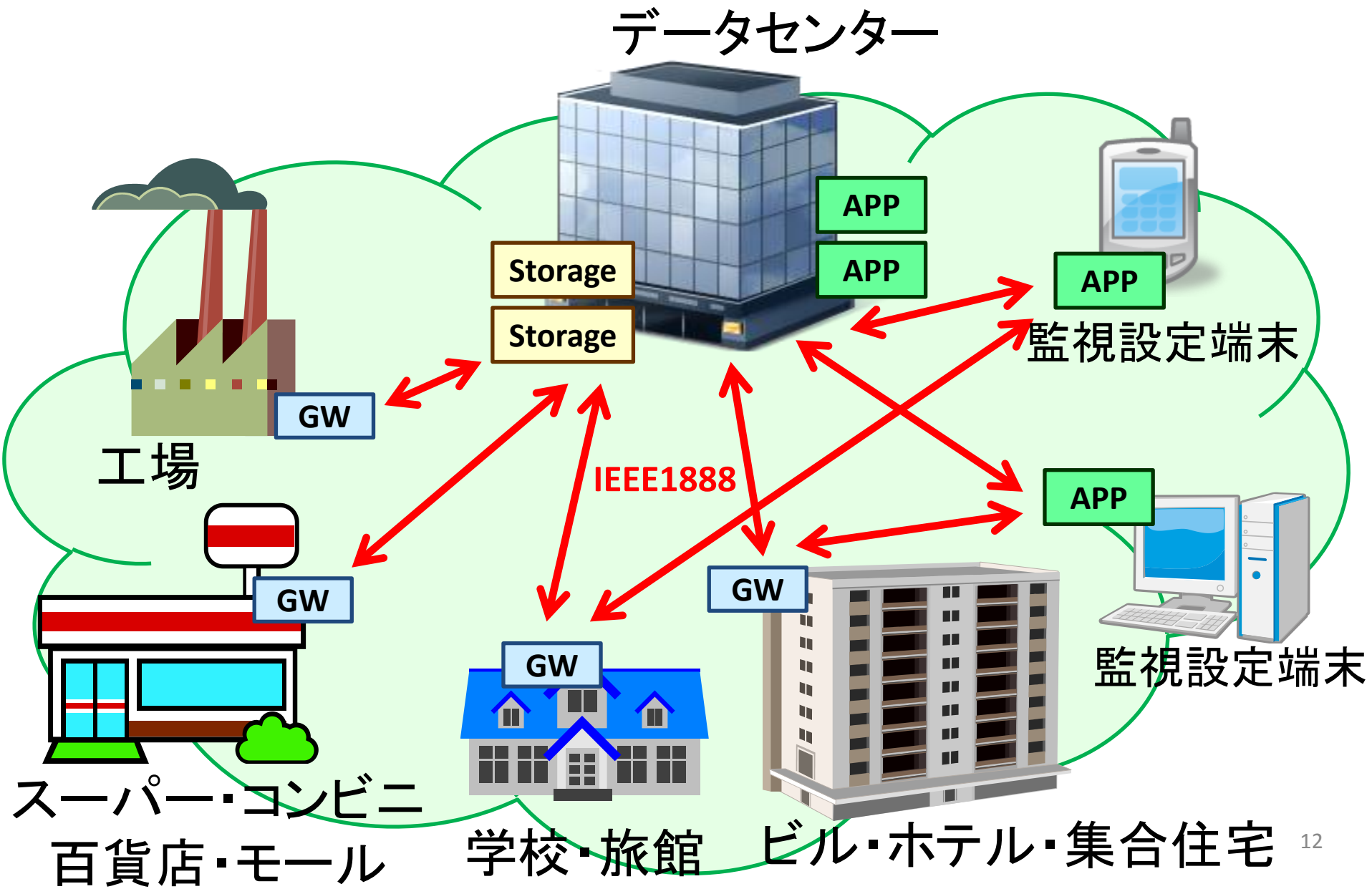
- ユビキタス時代の到来　(2010年代～)
  - スマートフォン、タブレット端末が普及した
  - クラウド(インターネットの向こう側)で、アプリケーションが動いている
  - 情報システム間は HTTP+XML によるオブジェクトの交換が主流になった
  - 多様な通信媒体 (例えば、無線センサネットワーク) が実用化された

➔ **このような変化に合わせた通信規格が必要となることに**

IEEE1888：HTTP+XMLで“設備機器”、“データベース”、“情報システム”をつなぐ通信プロトコル



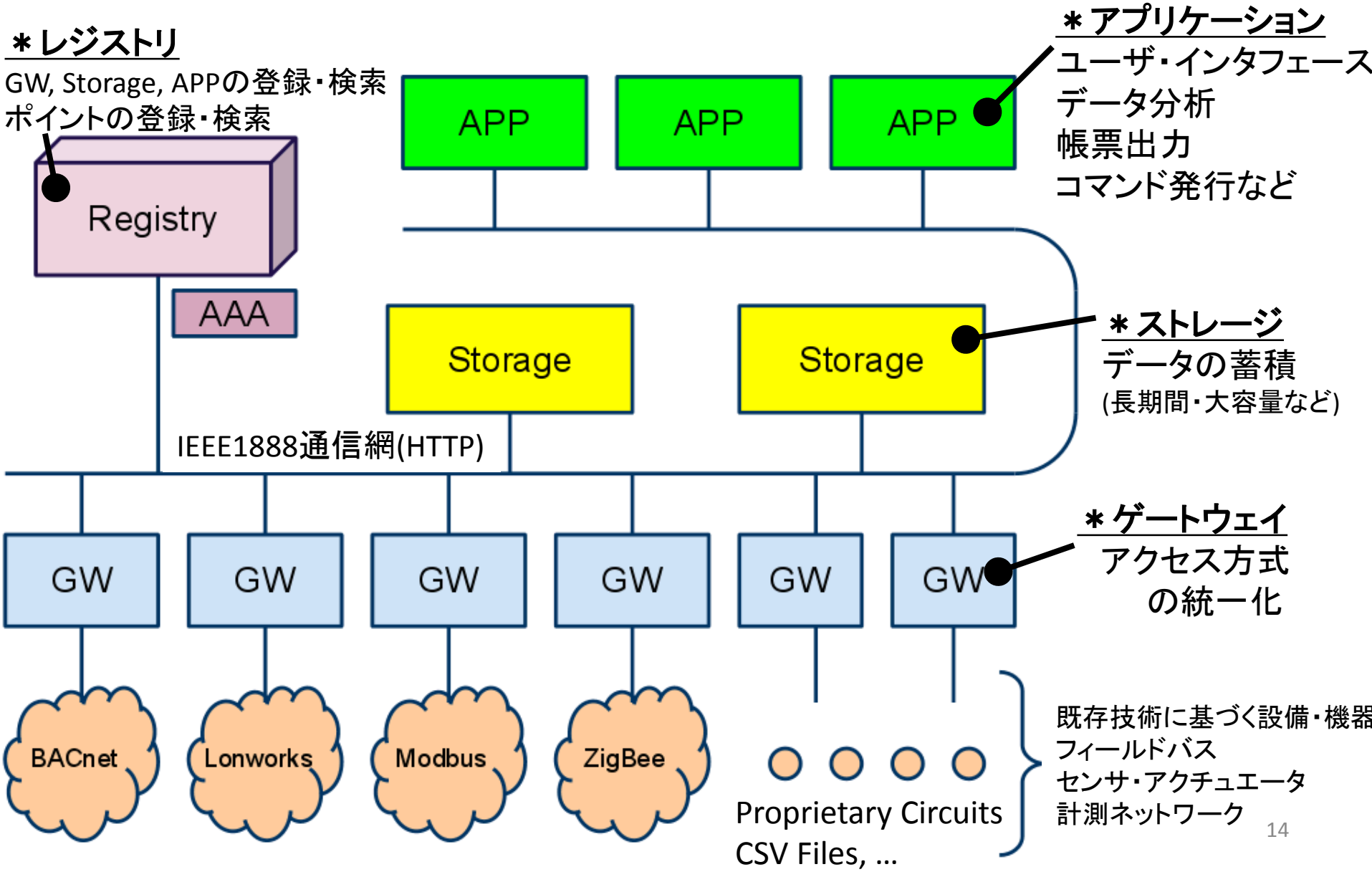
IEEE1888通信プロトコルの応用イメージ



IEEE1888 とは何か？

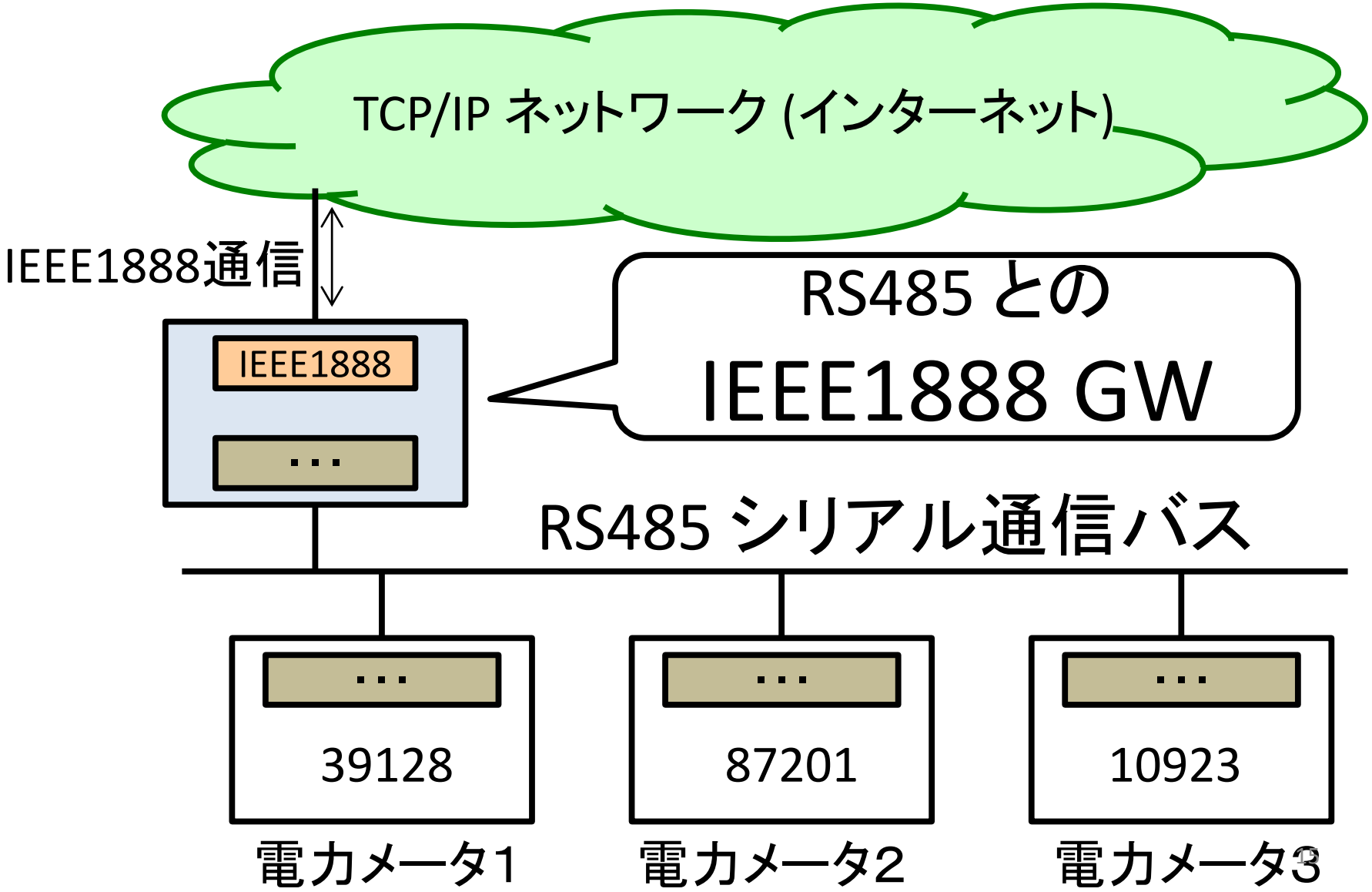
- 背景、目的、概要
- **アーキテクチャ**
  - **コンポーネント (GW, Storage, APP)**
  - **通信手順 (WRITE, FETCH, TRAP)**
  - **システムの実現方法**
- データ構造
  - メッセージ・フォーマット
  - “ポイント”の概念
  - 具体例
- 参照機器モデル
- 東大の5万kW電力需要家の管理事例

IEEE1888 システム・アーキテクチャ

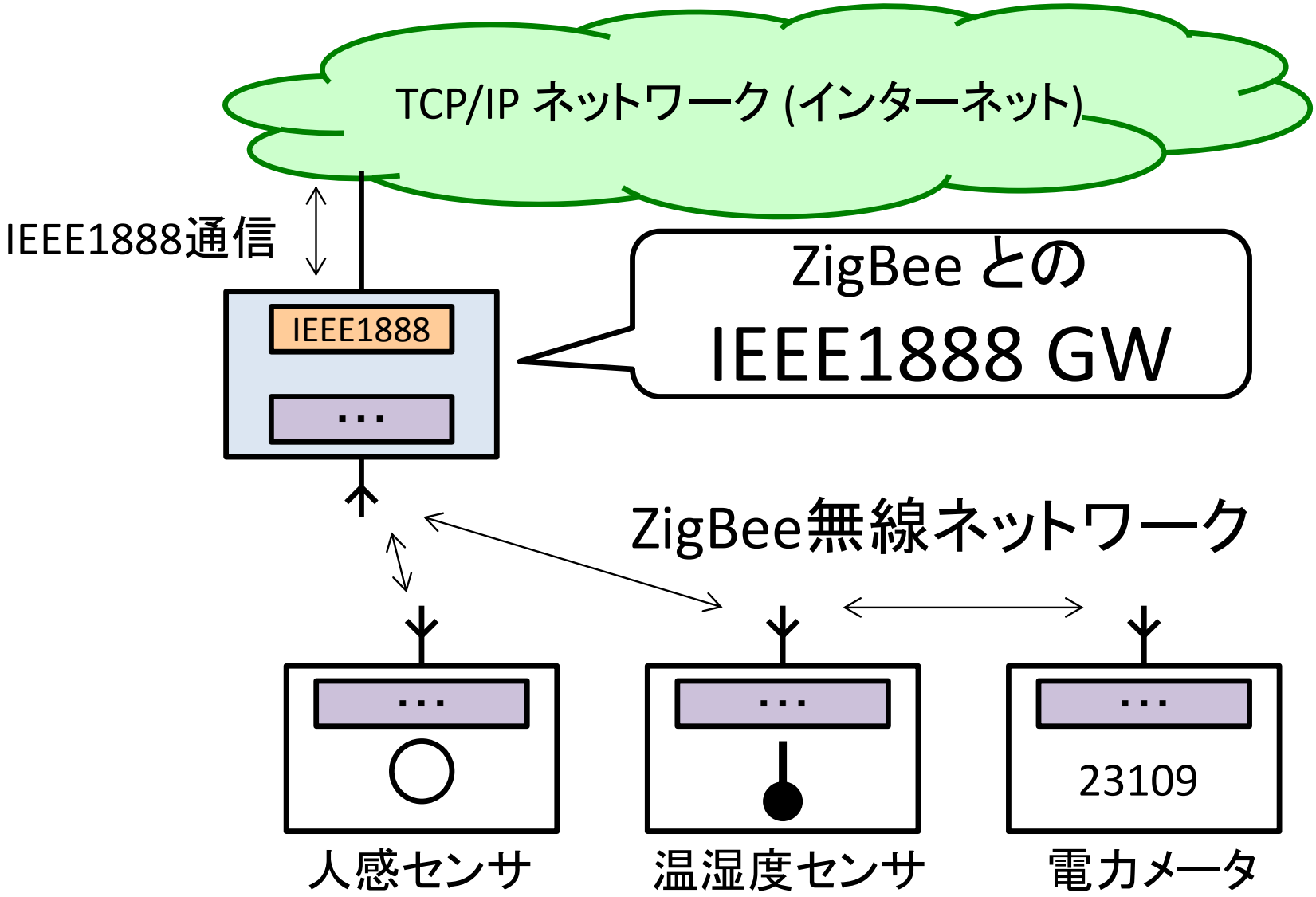


ゲートウェイ(GW)の例：その1

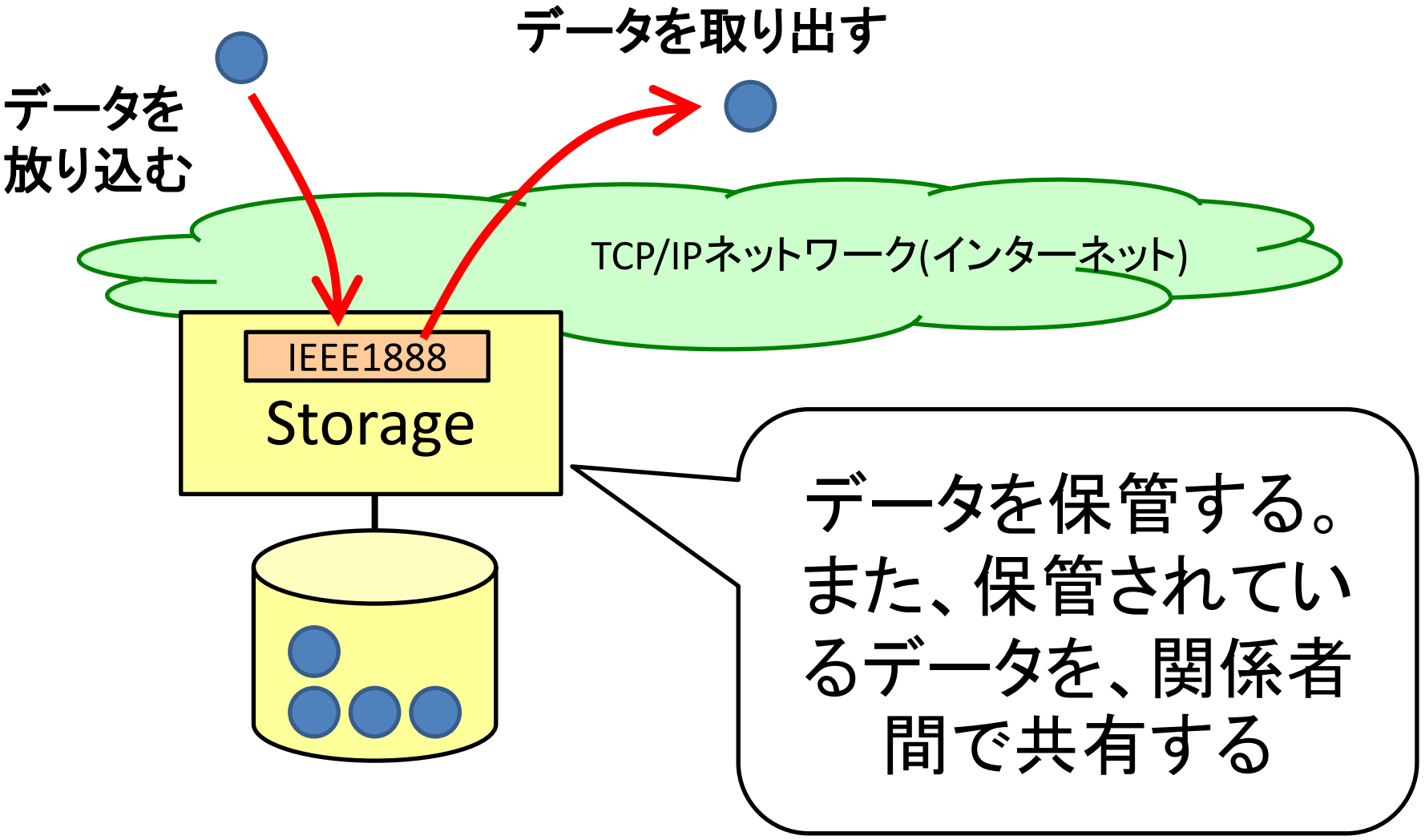
IEEE1888-RS485 GW装置



ゲートウェイ(GW)の例：その2
IEEE1888-ZigBee 変換GW

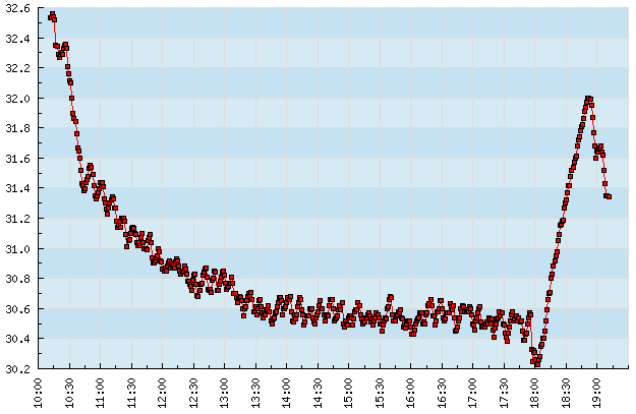


ストレージ(Storage)の例



アプリケーション（APP）の例：その１
見える化APP

IEEE1888サーバからデータを取得し、一覧で表示したり、過去データをグラフ化したりする機能を持つ。



PNG画像ファイルの生成

見える化APP

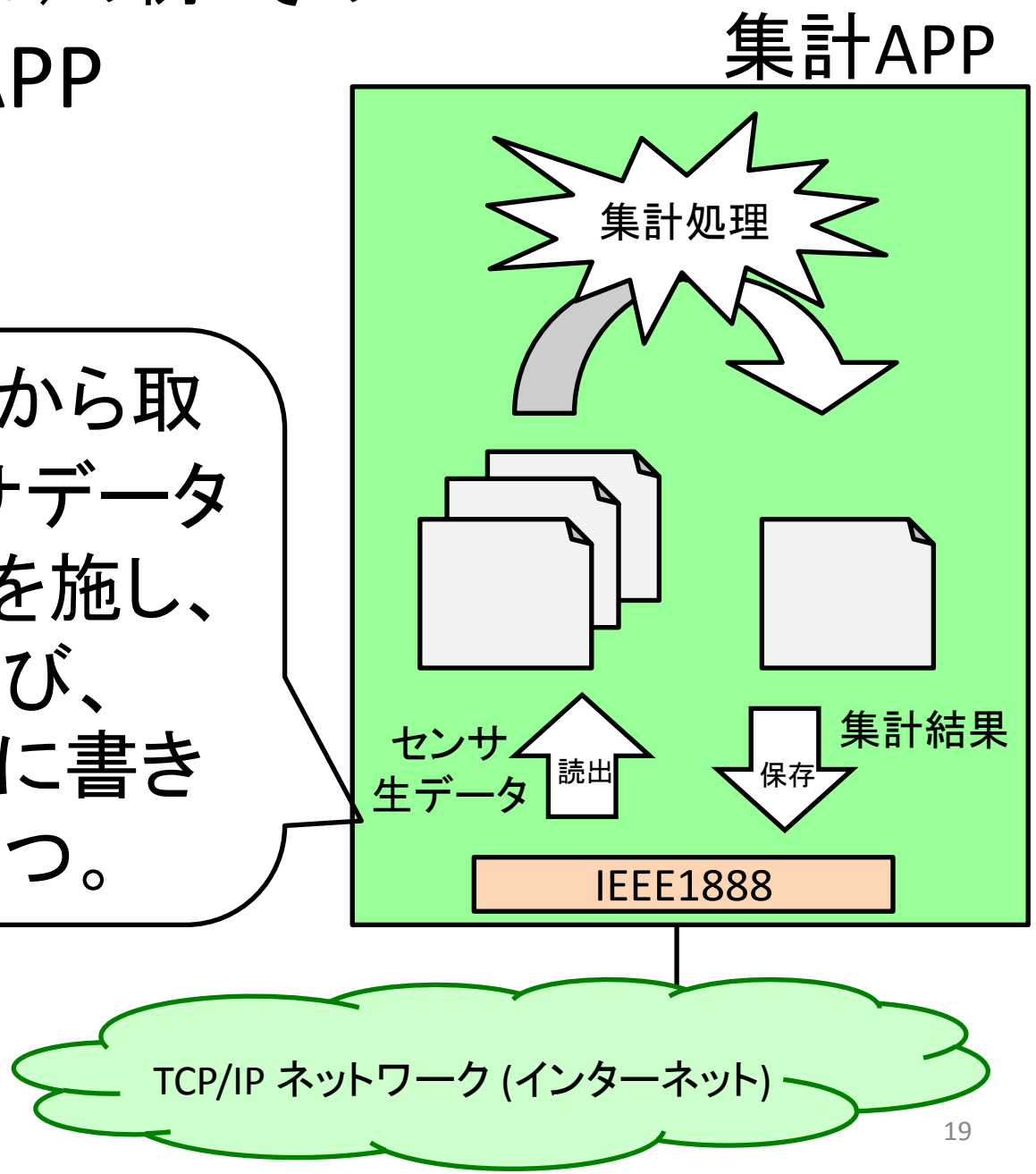
IEEE1888

TCP/IP ネットワーク (インターネット)

アプリケーション（APP）の例：その２

集計APP

IEEE1888サーバから取得した生のセンサデータに集計演算処理を施し、その結果を再び、IEEE1888サーバに書き出す機能を持つ。



整理すると・・・

- ゲートウェイ (GW)
  - 入出力(センサ・アクチュエータ)機器へのアクセスを IEEE1888方式に対応させる装置
- ストレージ (Storage)
  - IEEE1888でオンライン化されたデータを、保管および共有するための装置
- アプリケーション (APP)
  - IEEE1888でオンライン化されたデータと、ユーザとのインタフェースを担う装置
  - ユーザの望む形式に加工する装置

  など

(重要)これらは、あくまでも、便宜上の呼び名。プロトコル定義的には、GW，Storage，APPの間には違いは存在しない。

IEEE1888 コンポーネント

- GW, Storage, APPは総称して**IEEE1888 コンポーネント**と呼ばれ、すべて同じインタフェースを持つ。
  - queryメソッド
    - クエリ(取得要求)を投げ込むためのメソッド
  - dataメソッド
    - データ本体を投げ込むためのメソッド

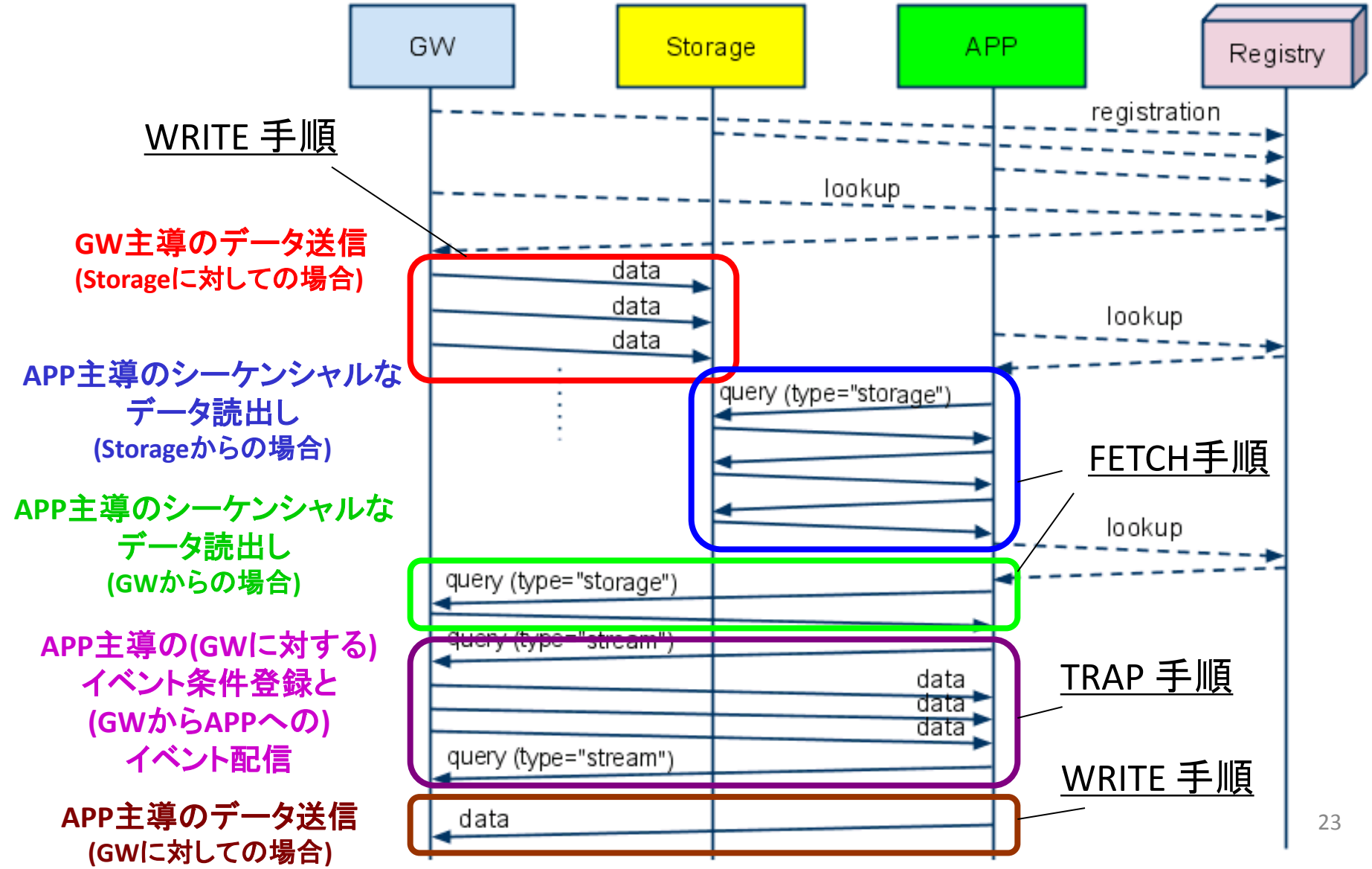
GW, Storage, App はすべてこのモデルに従う

これらメソッドの組合せで3つの通信手順が定義されている。

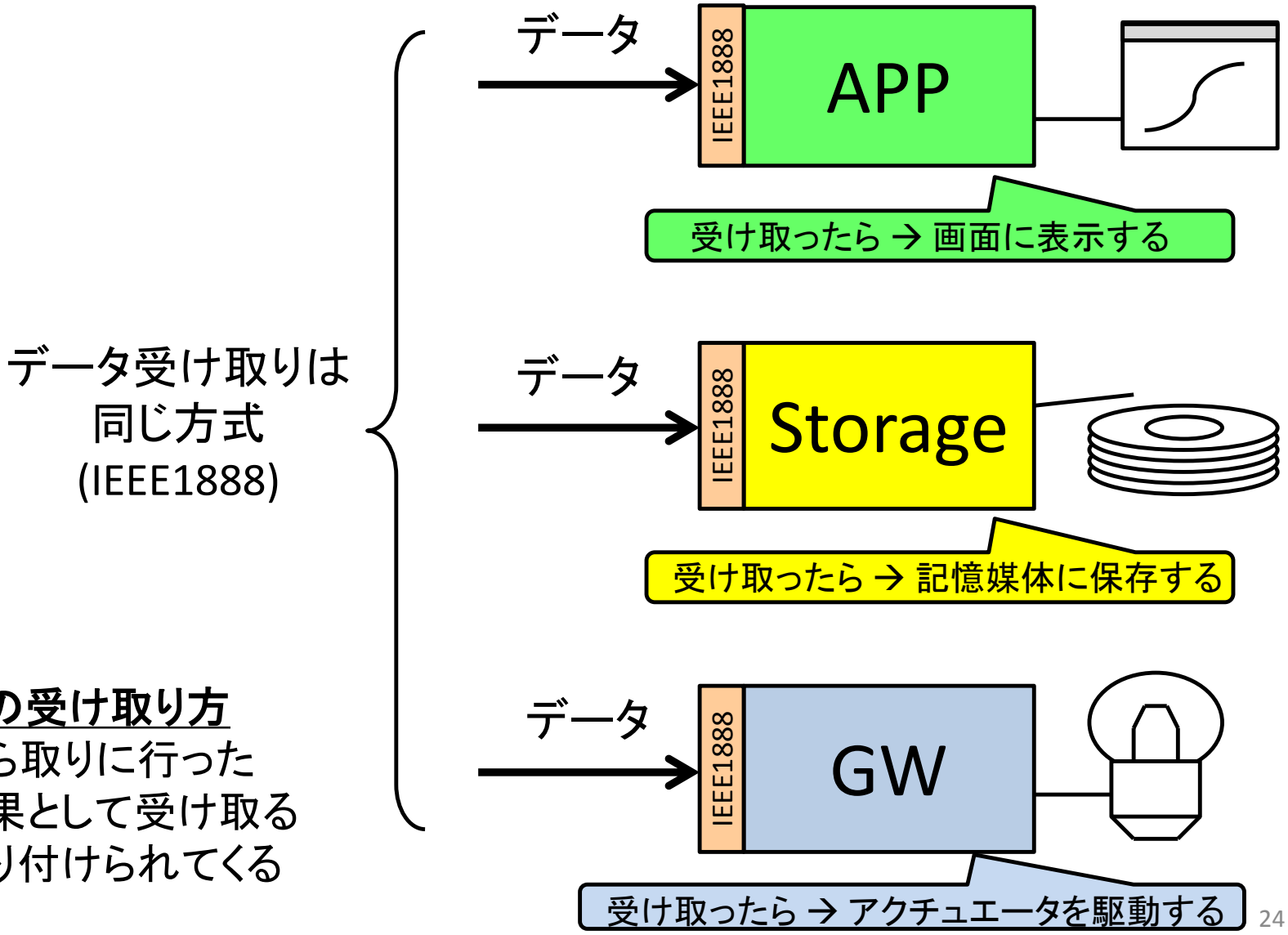
IEEE1888コンポーネント間通信 ３つの通信手順

- WRITE手順
  - データを対向のコンポーネントに送り込む手順
  - dataメソッドの読出し (1回)
- FETCH手順
  - 対向のコンポーネントから、今あるデータを抜出す手順
  - queryメソッドの呼出し (連続複数回)
- TRAP手順
  - 対向のコンポーネントから、将来発生するデータをもらう手順
  - queryメソッドの呼出し (定期的に実施)
  - dataメソッドの呼出しによるcallback (変化発生時)

IEEE1888コンポーネント間通信
WRITE, FETCH, TRAP手順のシーケンス例



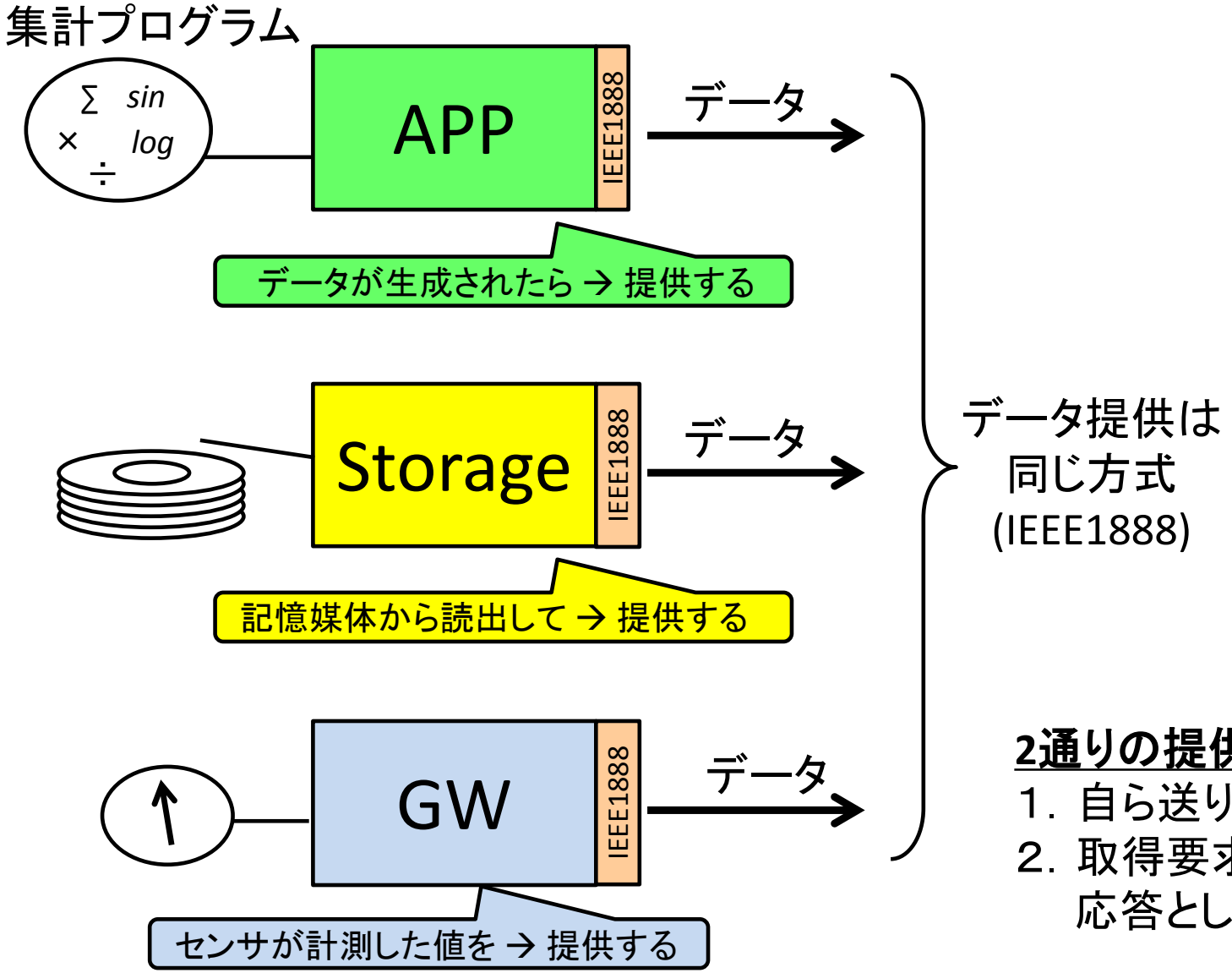
GW, Storage, APPの本質的な違い：その1



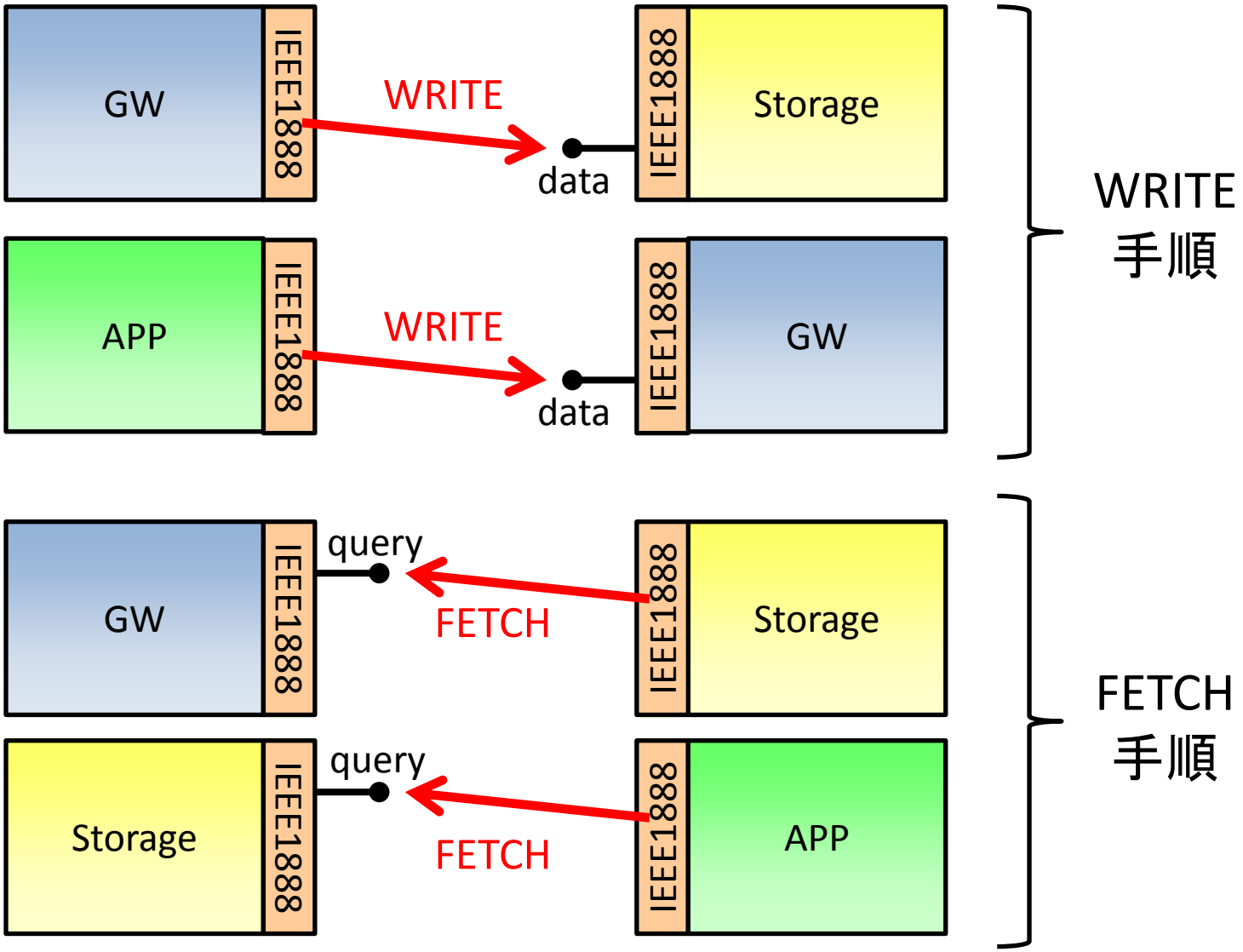
2通りの受け取り方

1. 自ら取りに行った結果として受け取る
2. 送り付けられてくる

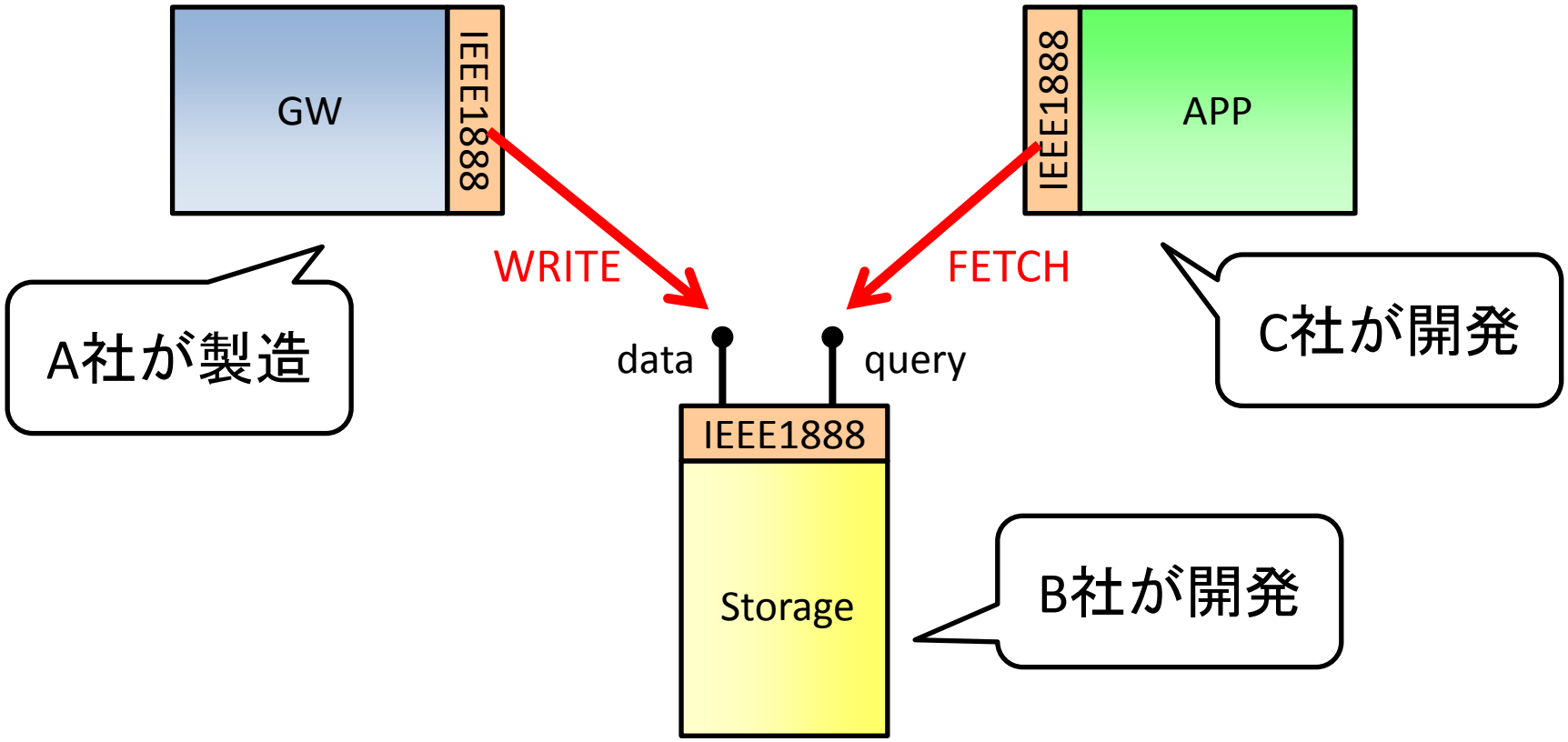
GW, Storage, APPの本質的な違い：その2



IEEE1888通信プロトコルは、
コンポーネント間をつないで、システムを組み上げる



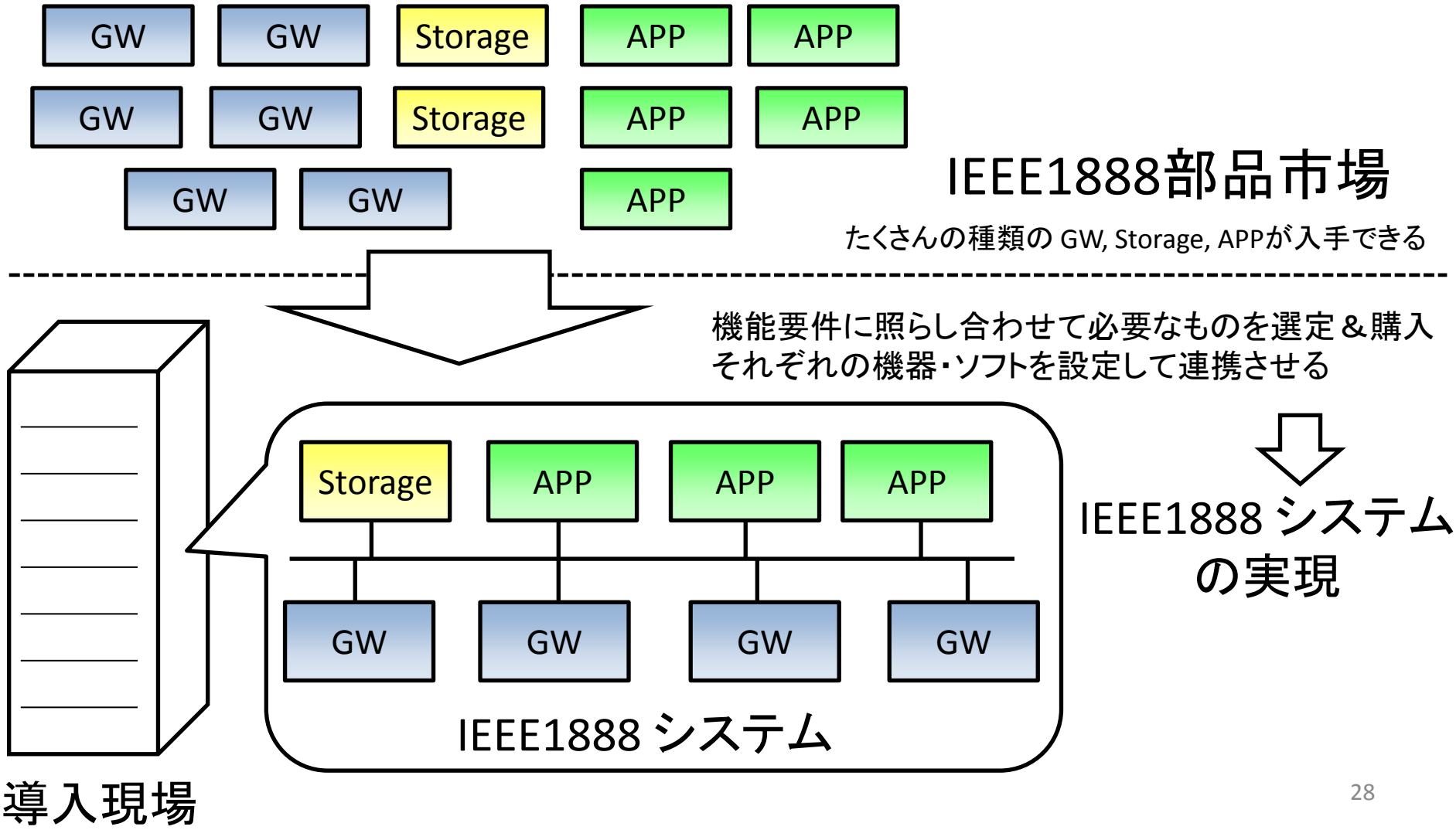
IEEE1888通信プロトコルは、
コンポーネント間をつないで、システムを組み上げる



GWによってオンライン化された「電力消費量」が、
Storageによって保管され、APPによって「見える化」される。

それぞれの部品(コンポーネント)は他社から提供されていても問題ない

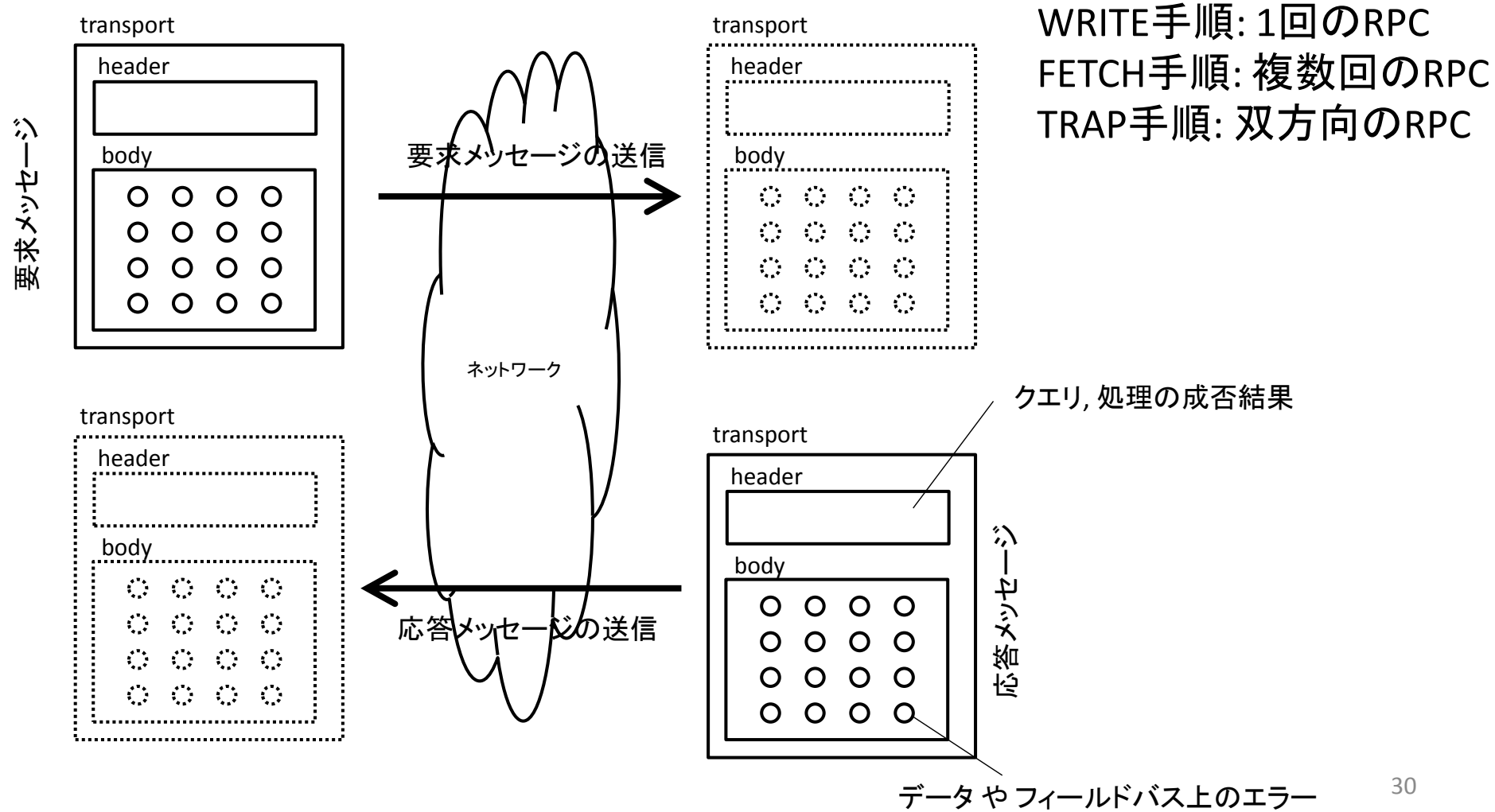
IEEE1888のシステム実現の考え方



IEEE1888 とは何か？

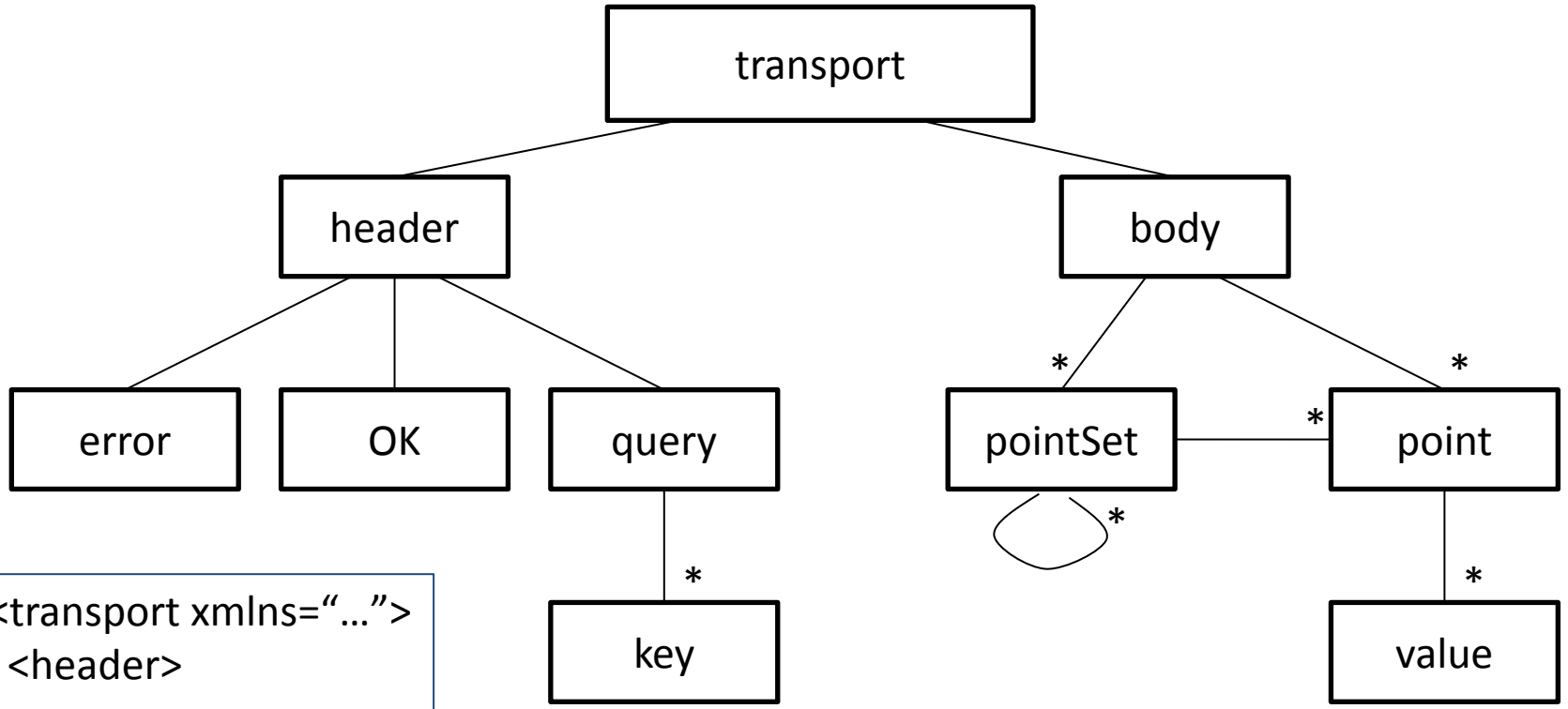
- 背景、目的、概要
- アーキテクチャ
  - コンポーネント (GW, Storage, APP)
  - 通信手順 (WRITE, FETCH, TRAP)
  - システムの実現方法
- **データ構造**
  - **メッセージ・フォーマット**
  - **“ポイント”の概念**
  - **具体例**
- 参照機器モデル
- 東大の5万kW電力需要家の管理事例

IEEE1888の通信メッセージは遠隔手続き呼出し(RPC: Remote Procedure Call)に埋め込む



WRITE手順: 1回のRPC
FETCH手順: 複数回のRPC
TRAP手順: 双方向のRPC

IEEE1888メッセージの構造




```
<transport xmlns="…">
  <header>
    ….
  </header>
  <body>
    ….
  </body>
</transport>
```

XMLへの展開例

* は複数個の要素を子として持つことを意味する

XML名前空間: http://gutp.jp/fiap/2009/11/

IEEE1888 メッセージ・フォーマット

WRITE(要求)の例

```
POST /axis2/services/FIAPStorage HTTP/1.1
Content-Type: text/xml; charset=UTF-8
SOAPAction: "http://soap.fiap.org/data"
User-Agent: Axis2
Host: 192.168.10.198

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<ns2:dataRQ xmlns:ns2="http://soap.fiap.org/">
 <transport xmlns="http://gutp.jp/fiap/2009/11/">
  <body>
   <pointSet id="http://jo2lxq.hongo.wide.ad.jp/test/">
    <point id="http://jo2lxq.hongo.wide.ad.jp/test/Temperature">
     <value time="2011-09-03T21:20:00.000+09:00">25.4</value>
     <value time="2011-09-03T21:21:00.000+09:00">25.5</value>
    </point>
    <point id="http://jo2lxq.hongo.wide.ad.jp/test/CO2">
     <value time="2011-09-03T21:20:00.000+09:00">541</value>
     <value time="2011-09-03T21:21:00.000+09:00">542</value>
    </point>
    <point id="http://jo2lxq.hongo.wide.ad.jp/test/Power">
     <value time="2011-09-03T21:20:00.000+09:00">6583.5</value>
     <value time="2011-09-03T21:21:00.000+09:00">6583.3</value>
    </point>
   </pointSet>
  </body>
 </transport>
</ns2:dataRQ>
</soapenv:Body>
</soapenv:Envelope>
```

※ このURIは<u>ポイントID</u>と呼ばれる

（ここにアクセスしてもデータが得られるわけではない）

温度センサのデータ

CO2センサのデータ

電力センサのデータ

IEEE1888サーバアドレス(SOAP通信端点)

http://192.168.10.198/axis2/services/FIAPStorage

に送信されている (HTTPヘッダ部を参照)

IEEE1888メッセージ・フォーマット
FETCH応答例

```
-- FETCH最新値レスポンス例(ここから)--
<transport xmlns="http://gutp.jp/fiap/2009/11/">
 <header>
  <OK/>
  <query id="0a90f1fa-bdb4-48ff-87d3-661d2af6ff4c" type="storage">
   <key id="http://www.gutp.jp/dummy/integer" attrName="time" select="maximum"/>
   <key id="http://www.gutp.jp/dummy/real1" attrName="time" select="maximum"/>
  </query>
 </header>
 <body>
  <point id="http://www.gutp.jp/dummy/integer">
   <value time="2011-09-03T11:35:00.000+09:00">76419323</value>
  </point>
  <point id="http://www.gutp.jp/dummy/real1">
   <value time="2011-09-03T11:35:00.000+09:00">-99.8</value>
  </point>
 </body>
</transport>
-- FETCH最新値レスポンス例(ここまで)--
```

要求の成功・失敗 に関する記述

データに関する記述

(FETCHの場合は読み出されたデータ)

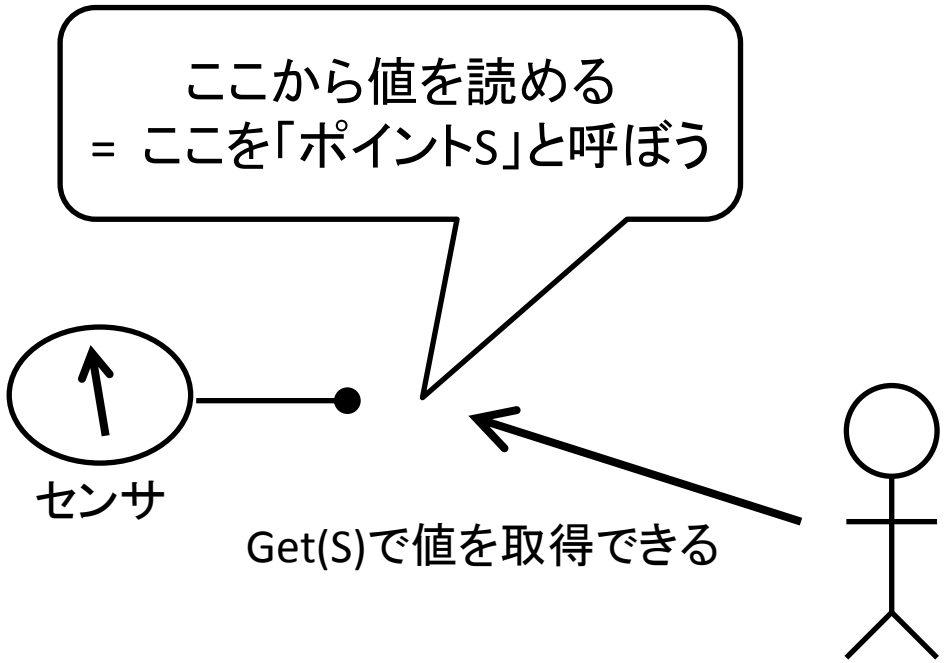
“ポイント”の概念：従来の考え方

センサ・アクチュエータなどに関連付けられるインタフェースを

ポイント

と呼び、それぞれにIDをつけた。

そのIDに対してGET, SET等のアクションを起こすことで、
1．センサデータの取得
2．アクチュエータ制御
ができた。



ID = 場所(Location)を表す識別子
でもあった

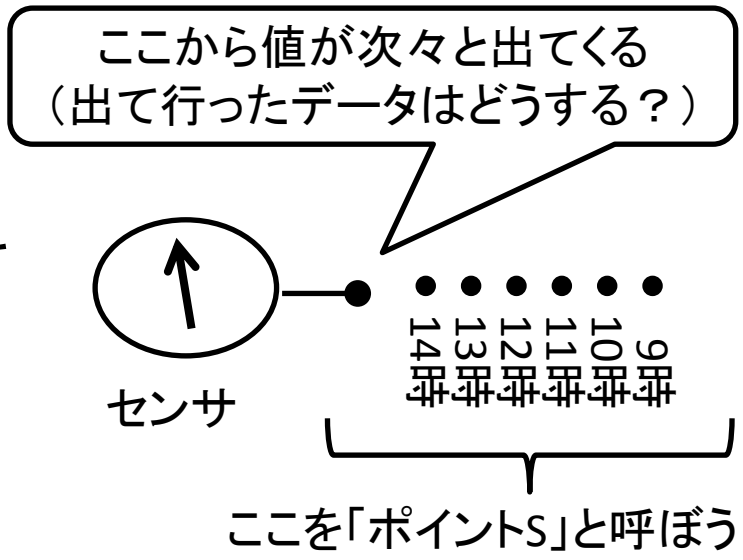
“ポイント”の概念：IEEE1888では

IEEE1888では、時系列データの
管理＆扱い方式が強化されている。

センサ・アクチュエータなどに関連付けられる時系列データを

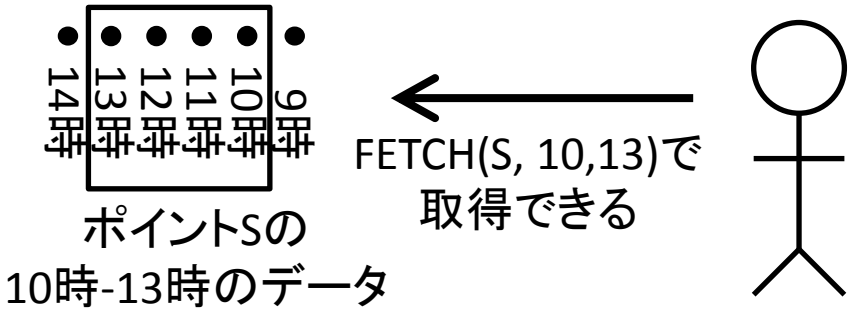
ポイント

と呼び、そこにIDをつける。

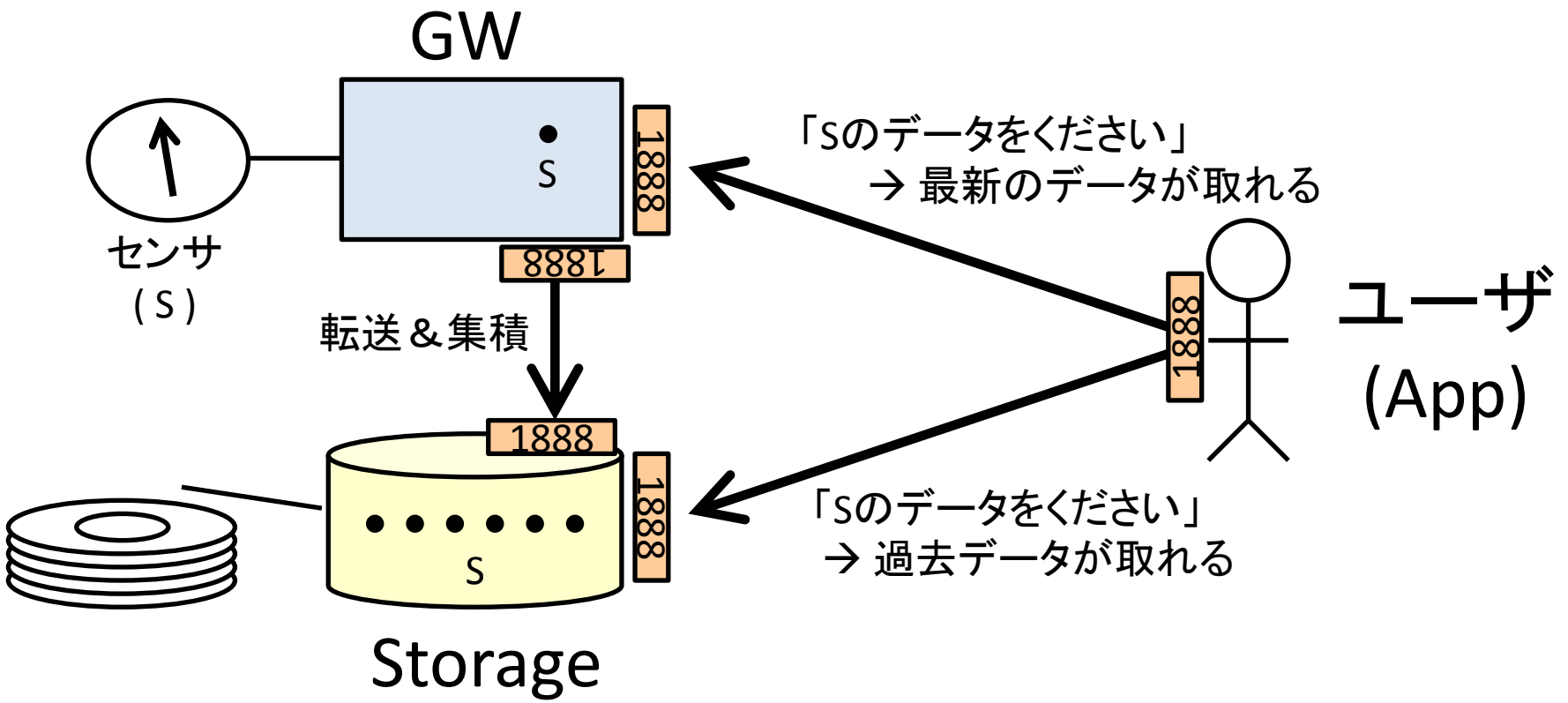


データの保管場所は、IDのみでは
決まらない。

ID と データのある場所は、
別々に管理される(次ページ)。



“ポイント”の概念：IEEE1888では (つづき)



もし、ポイントIDが場所を表していたら、このような運用はできない

GW：センサからの値の取込みに特化
Storage: 過去データの保管に特化
ユーザ(App): 場所を切り替えるだけで同じようにデータを取得できる

“ポイント”の表記＆意味＆ポイント集合

- ポイントIDの表記
  -- グローバルに一意なURI であること

(推奨形式) http:// …. 管理組織 (またはGW)のドメイン名…. / 自由フォーマット

  - http://gutp.jp/Arduino/test-001/Temperature
  - http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/10F/102B1/Incoming_PPE

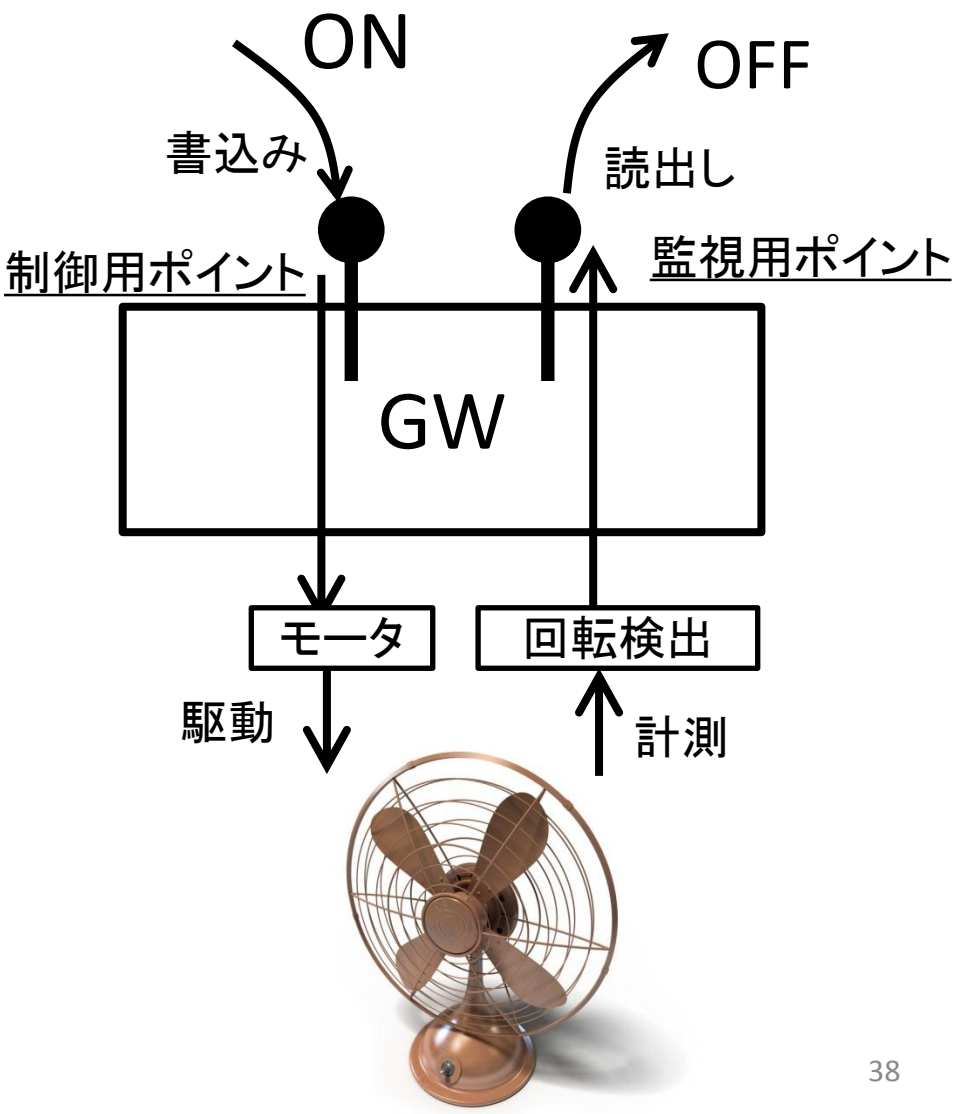
- ポイントIDへの意味づけ
  - ポイント表(明日の午後に解説)を使って管理するのが基本
  - ポイントIDそのものは意味を持たない (ルールを定めて持たせるのは可)

- ポイント集合(ポイントセット)
  - 複数のポイントをまとめて管理するために、ポイント集合の仕組みが定義されている
  - 例えば、「部屋」のようなまとまりを表すポイント集合を規定できる

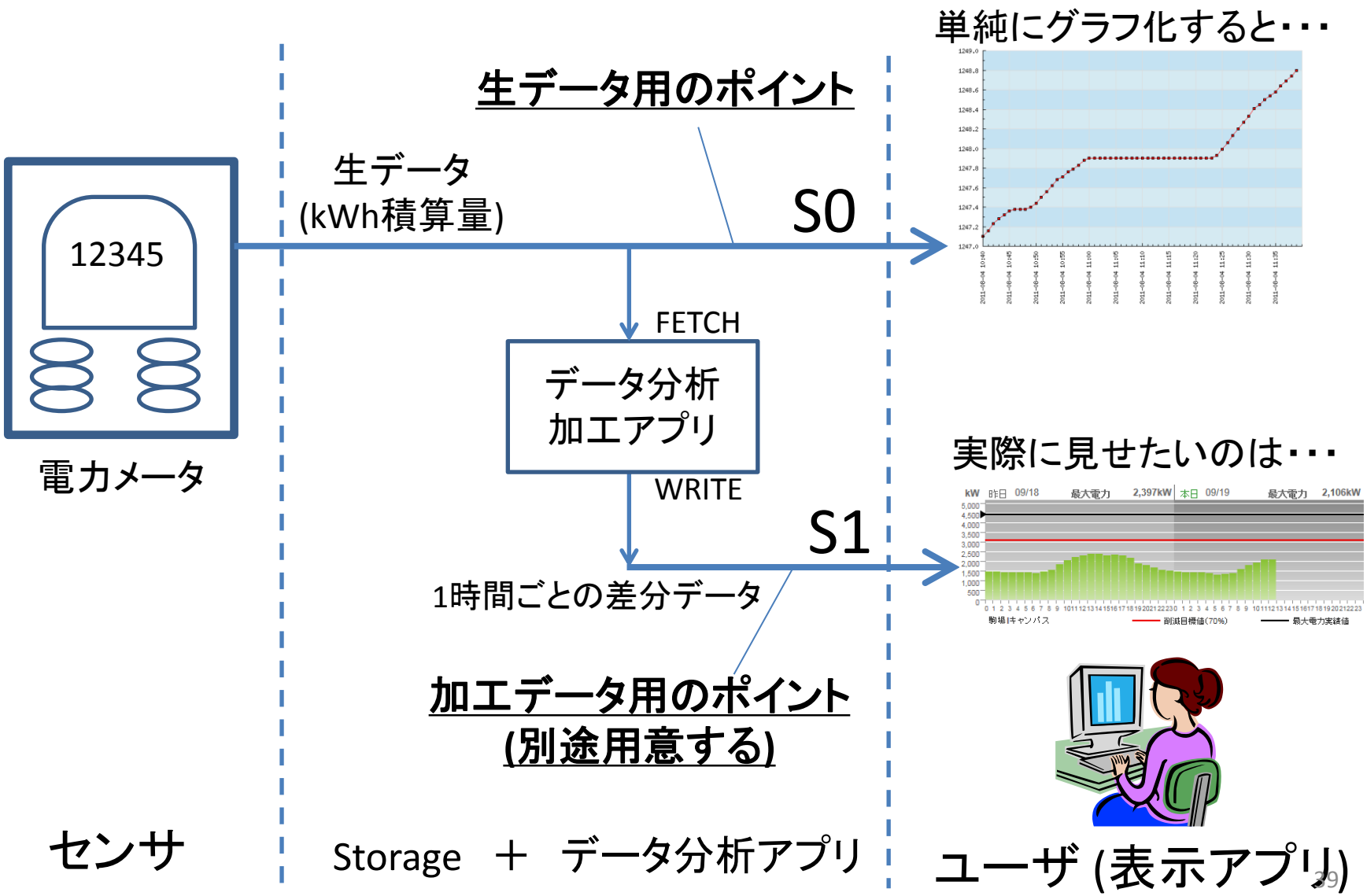
実際のポイント設計：その1
監視・制御の分離

同一対象でも、制御信号を輸送するポイントと、監視信号を輸送するポイントを別々に定義することは多い。

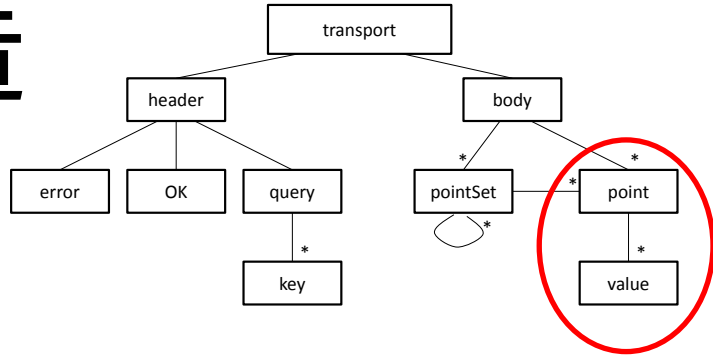
- 制御用のポイント
  - OnOffCtrl
- 監視用のポイント
  - OnOffStat



実際のポイント設計：その2
生データ・加工データの分離



“データ”のデータ構造



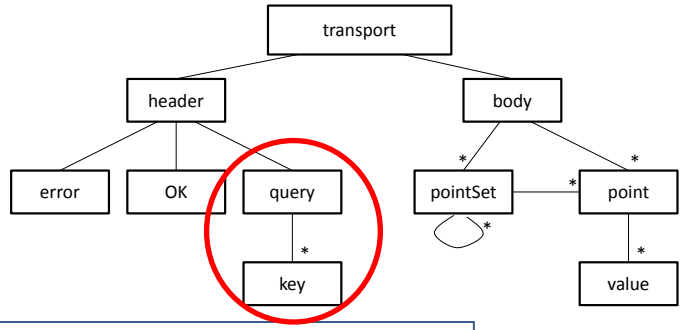
※ ポイントが時系列データを持つ

```
<point id=“http://gutp.jp/pointA”>
 <value time=“2012-01-01T00:00:00+09:00”>35.5</value>
 <value time=“2012-01-01T00:10:00+09:00”>35.5</value>
 <value time=“2012-01-01T00:20:00+09:00”>35.4</value>
 <value time=“2012-01-01T00:30:00+09:00”>35.3</value>
 <value time=“2012-01-01T00:40:00+09:00”>35.2</value>
 <value time=“2012-01-01T00:50:00+09:00”>35.1</value>
 <value time=“2012-01-01T01:00:00+09:00”>34.9</value>
</point>
```

時刻情報を持たない下記も可能
(ただし、実装によっては受け付けない場合もある)

```
<point id=“http://gutp.jp/pointA”>
 <value>35.5</value>
</point>
```

“クエリ”のデータ構造




```
<query id="…UUID…" type="storage" acceptableSize="100">
 <key id="http://gutp.jp/pointA" attrName="time"
        gteq="2012-01-01T00:00:00+09:00"
            lt="2012-02-01T00:00:00+09:00" />
</query>
```

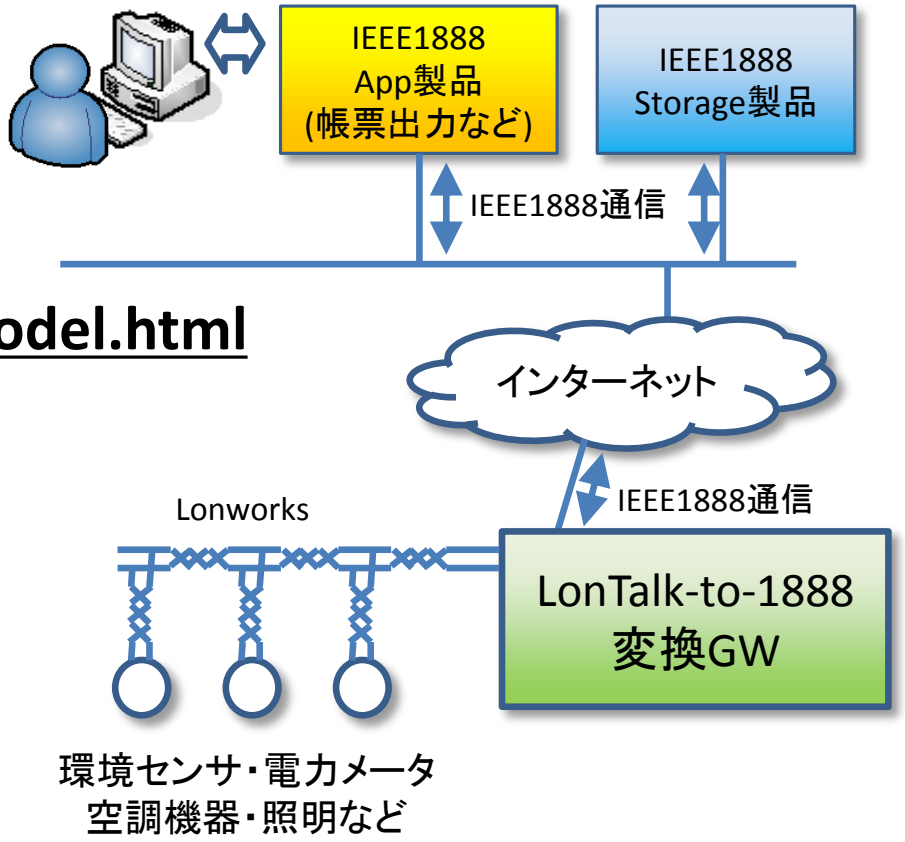
```
<query id="…UUID…" type="storage" acceptableSize="100">
 <key id="http://gutp.jp/pointA" attrName="time" select="maximum" />
 <key id="http://gutp.jp/pointB" attrName="time" select="maximum" />
 <key id="http://gutp.jp/pointC" attrName="time" select="maximum" />
</query>
```

```
<query id="…UUID…" type="stream" ttl="120"
          callbackData="http://133.11.168.3/axis2/services/FIAPStorage">
 <key id="http://gutp.jp/pointA" attrName="time" trap="changed" />
</query>
```

IEEE1888 とは何か？

- 背景、目的、概要
- アーキテクチャ
  - コンポーネント (GW, Storage, APP)
  - 通信手順 (WRITE, FETCH, TRAP)
  - システムの実現方法
- データ構造
  - メッセージ・フォーマット
  - “ポイント”の概念
  - 具体例
- **参照機器モデル**
- **東大の5万kW電力需要家の管理事例**

IEEE1888 参照機器モデル



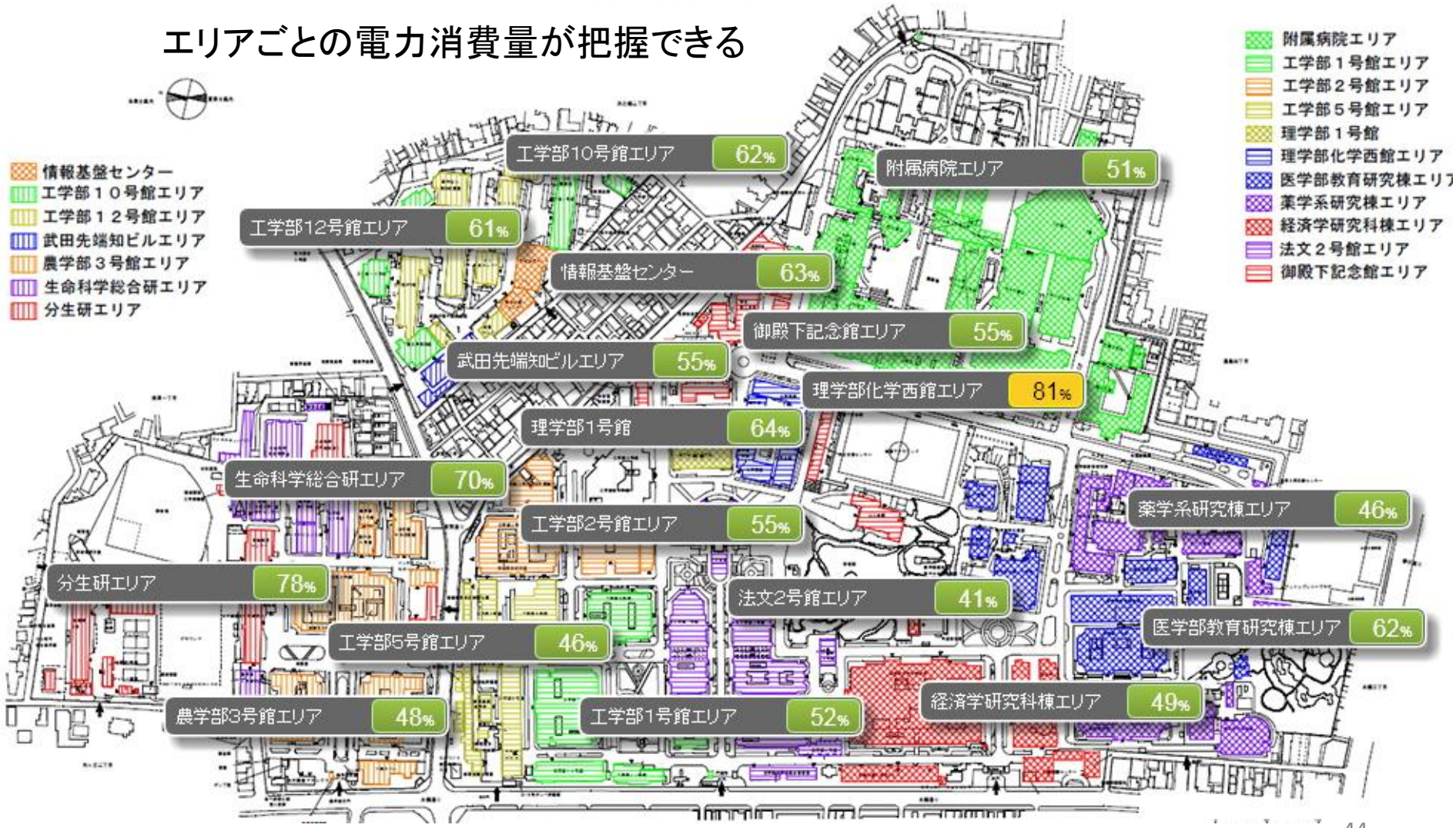
http://gutp.jp/fiap/model.html

- **GW編**
  - LonTalk-to-1888変換GW
  - BACnet-to-1888変換GW
  - ZigBee-to-1888変換GW
  - SNMP-to-1888変換GW
  - CSV-to-1888変換GW
  - IEEE1888汎用観測GW
- **Storage編**
  - 大容量ハイパフォーマンスStorage
  - ハイレベル分析機能搭載Storage
  - 小規模オフィス環境向けStorage
  - IEEE1888 Storageサービス
- **アプリ編**
  - 簡易監視(SCADA)アプリケーション
  - データ加工分析アプリケーション
  - ヘルスチェック・アプリケーション
  - MS Excel 連携アプリケーション
  - 電力デマンド制御アプリケーション

東大の事例

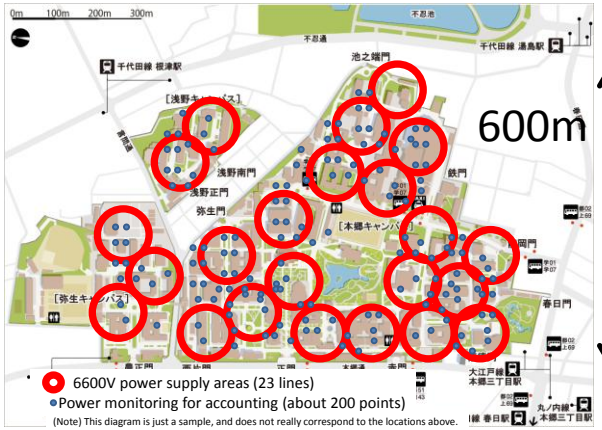
http://ep-monitor.adm.u-tokyo.ac.jp/areamap/

エリアごとの電力消費量が把握できる



東大の事例
(本郷キャンパスの場合)

1200m



キャンパスへの電力供給

30,000 kW 特高受電設備
(66kV → 6600V)

Storage

監視App
(Tool)

見える化App
(Tool)

IEEE1888

IEEE1888

IEEE1888

The Internet

IEEE1888

Gateway

(本郷キャンパスGW)

CSV

CSV

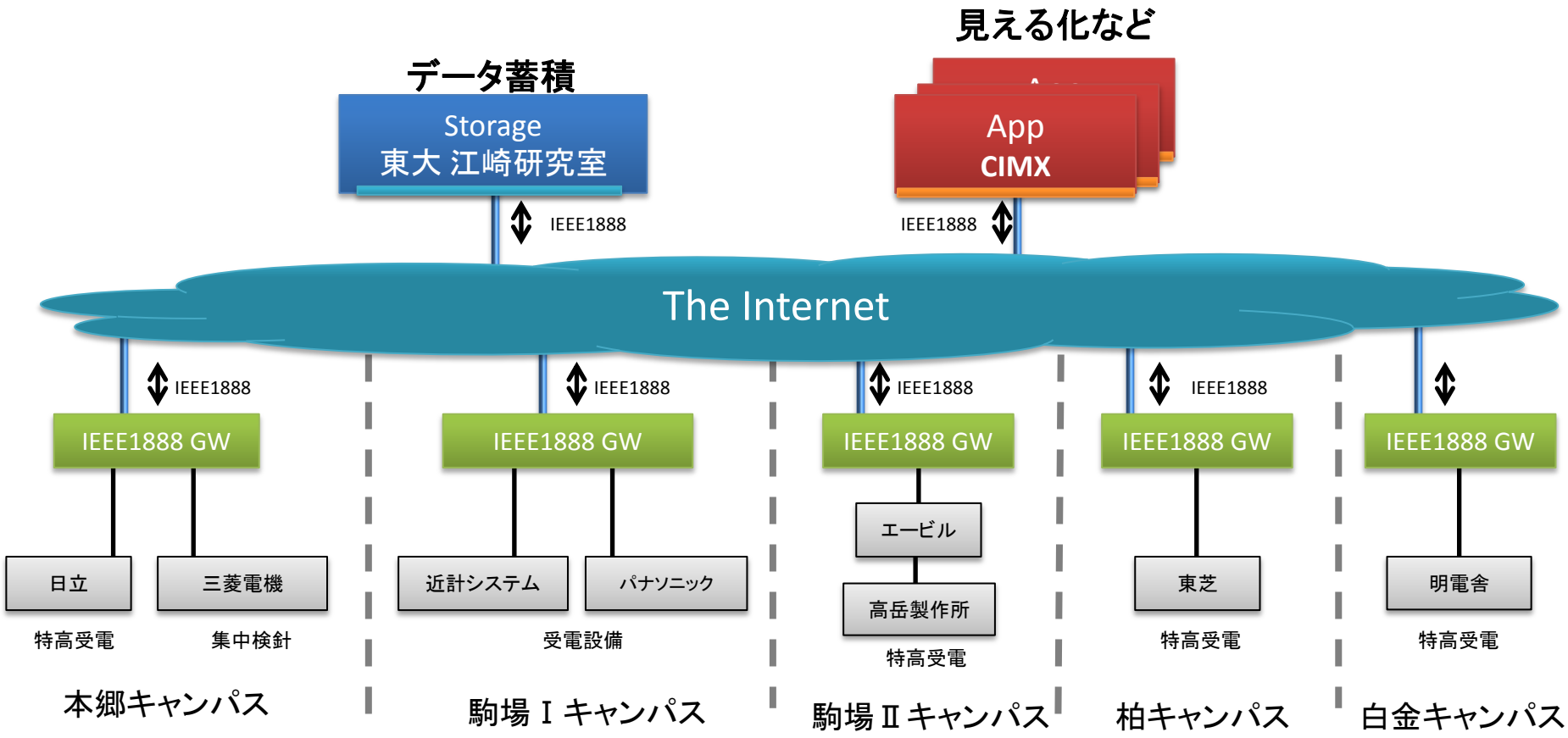
日立

三菱電機

BNET

Smart Meters
(100V, 200V)

東大の事例
(5キャンパスの統合管理)



合計で5万kWを超える電力が，管理対象(=監視制御対象)となっている
グラフ化や地図へのマッピング，警報発令などがIEEE1888によるITシステムで実現されている　http://ep-monitor.adm.u-tokyo.ac.jp/campus/denryoku

本日(15日)のスケジュール

10:00　IEEE1888とは何か？（全体像）

11:00　IEEE1888機器を使ってみる

12:00　昼食

13:00　IEEE1888ソフトウェア開発キット(SDK)の概要

13:30　SDKを使った FETCH手順のプログラミング

15:00　休憩

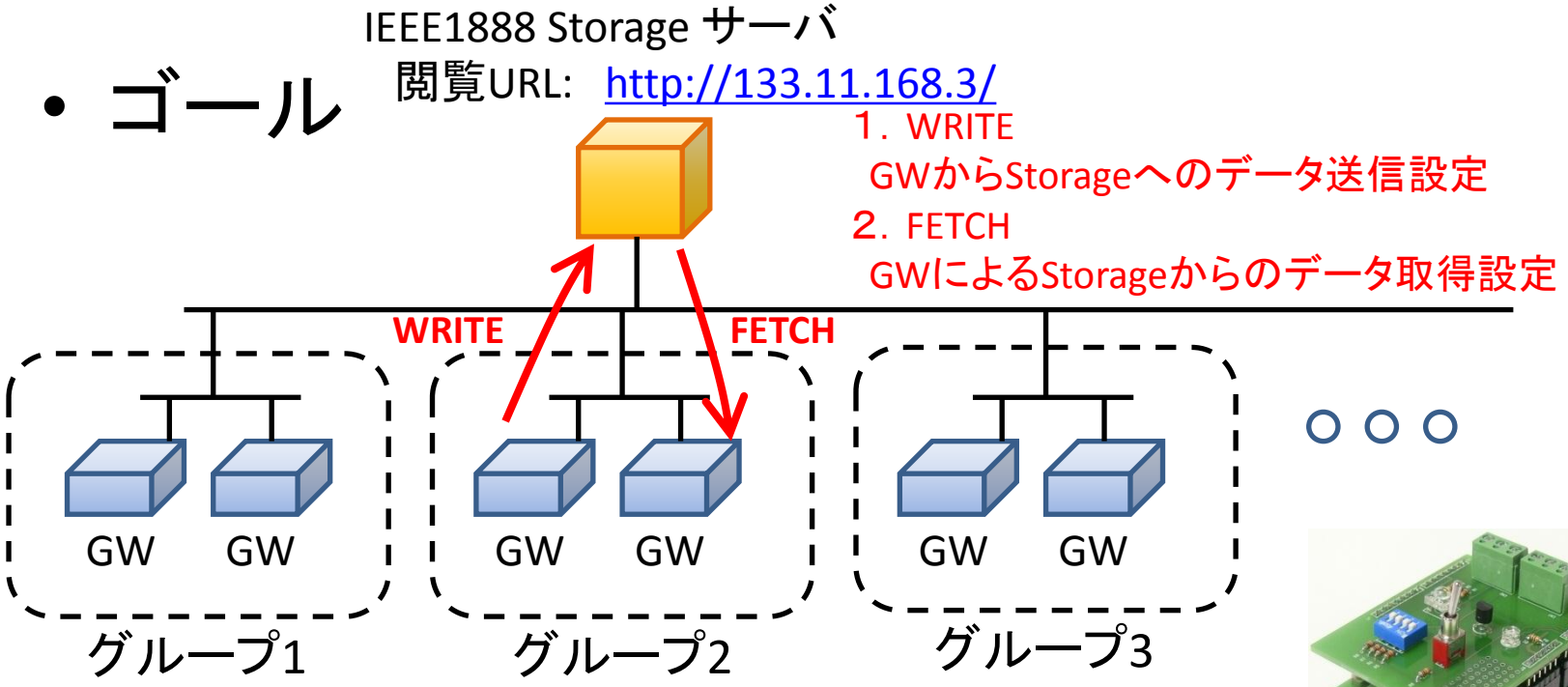
15:30　SDKを使った WRITE手順のプログラミング

16:30　C言語版/Arduino版についての紹介

17:00　解散

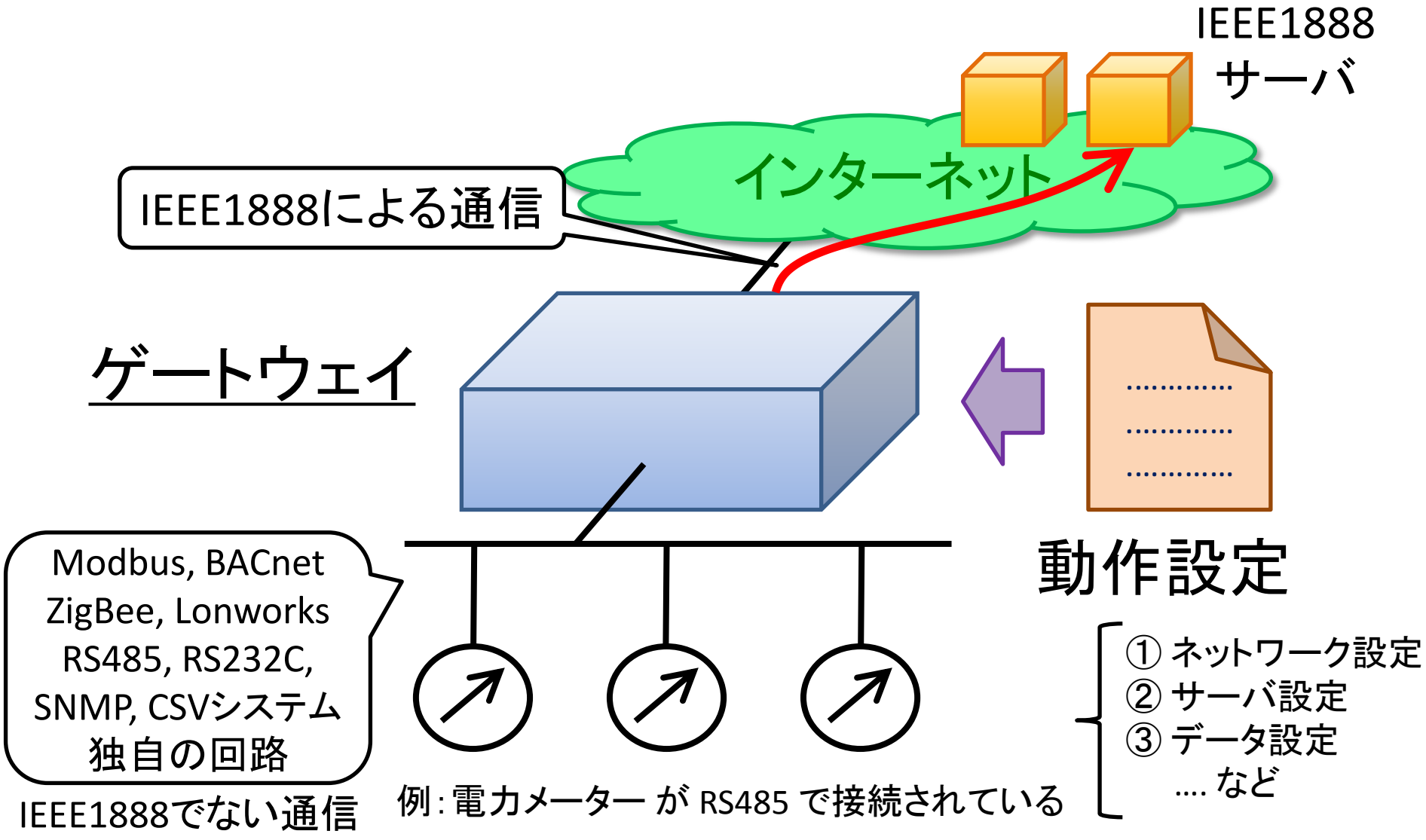
IEEE1888機器を使ってみる

- ゴール



- 進行方法
  - (1) GWのアーキテクチャを確認
  - (2) GWの設定と動作について
  - (3) 実践

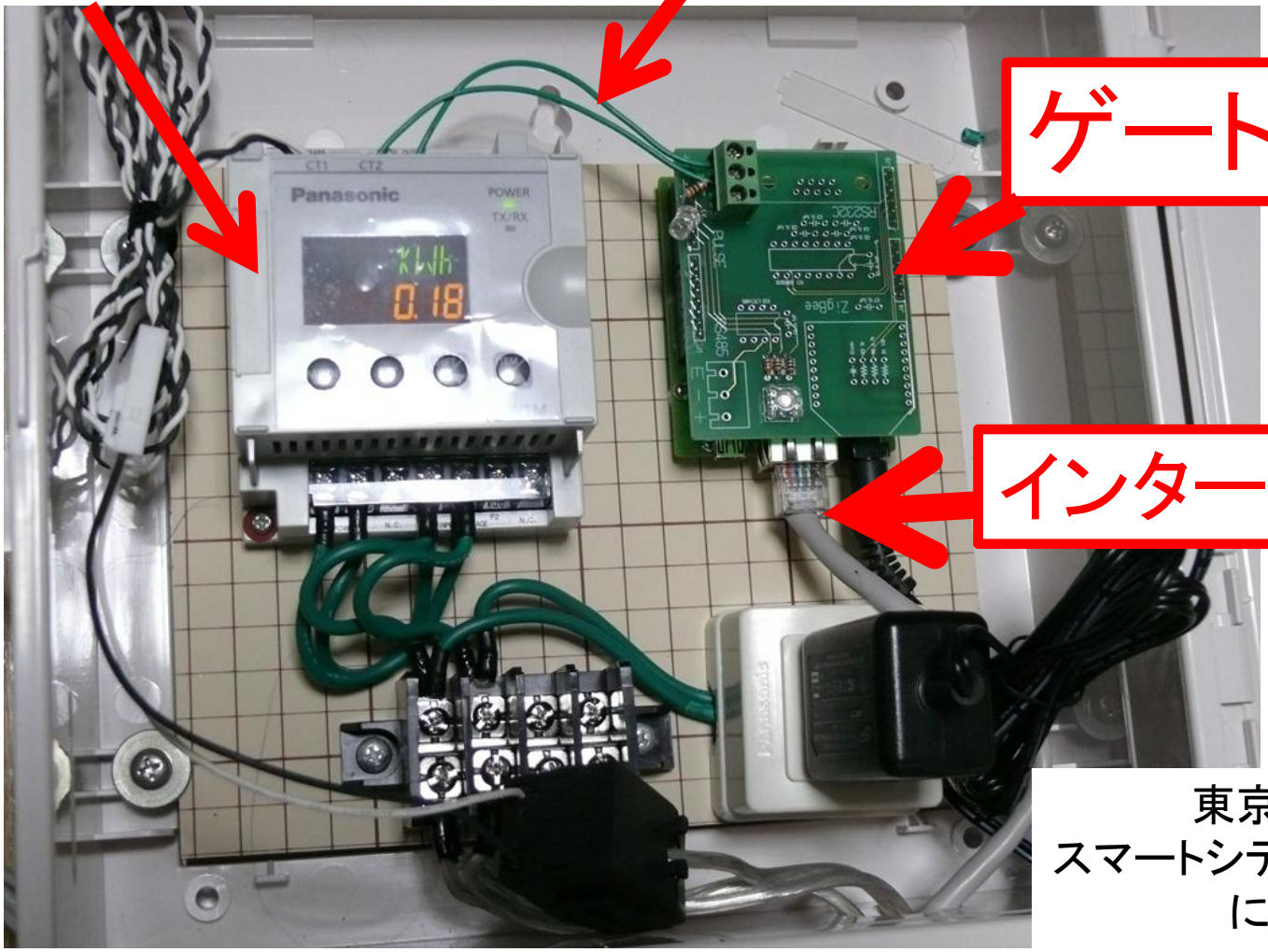
復習: GWのアーキテクチャ



<< IEEE1888機器を使ってみる >>

GWの実例1 (パルス信号)

Panasonic電力メータ　パルス信号出力(1Wh単位)



ゲートウェイ

インターネット接続

東京都・長野県
スマートシティ関連プロジェクト
にて運用中

<< IEEE1888機器を使ってみる >>

GWの実例2 (ZigBee通信)

IEEE1888通信 (FETCH, WRITE双方向)

スマートタップGW

首都大学
スマートオフィス関連研究プロジェクト
にて運用中

高精度スマートタップ

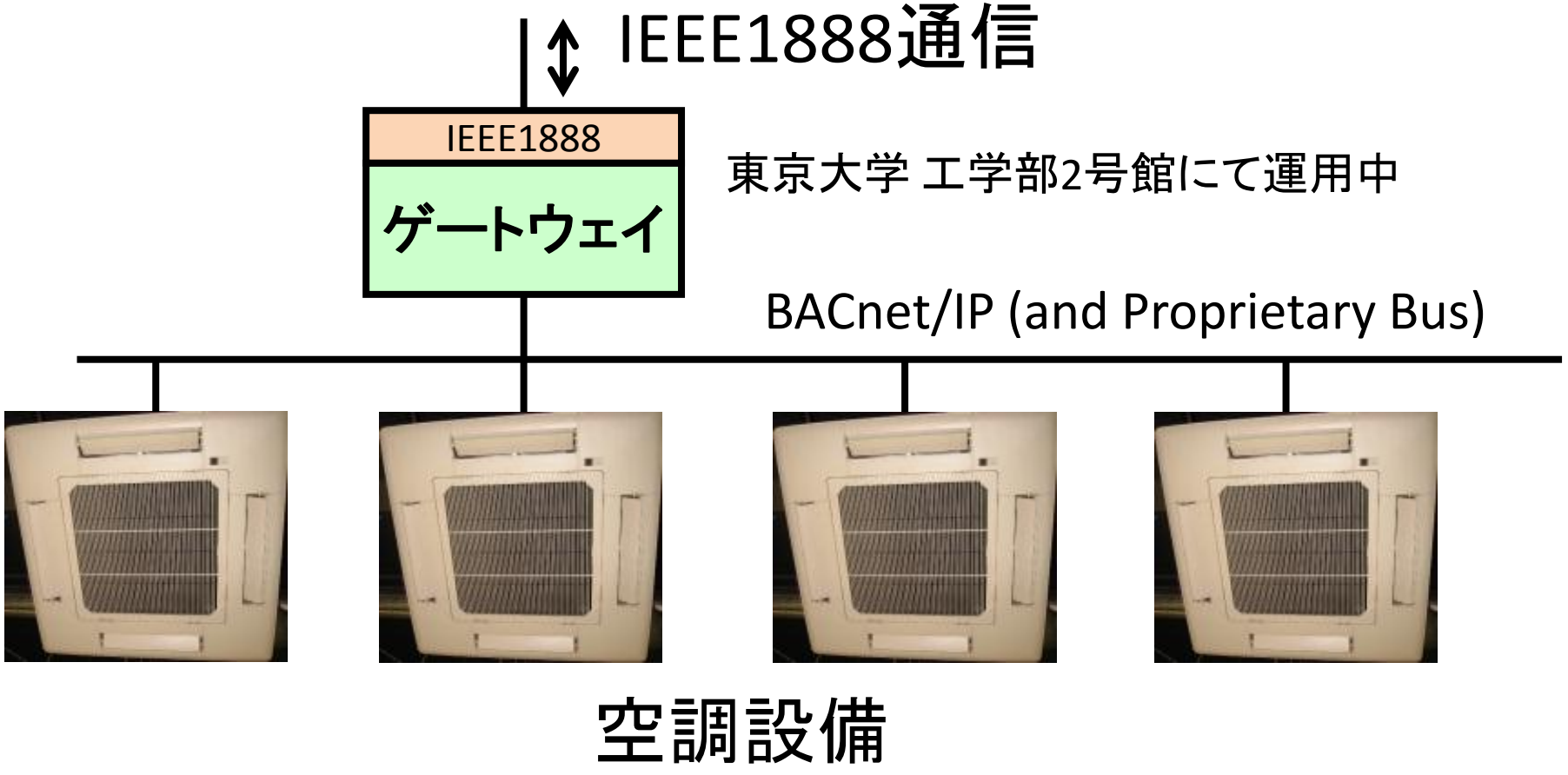
ZigBeeメッシュ
ネットワーク

高精度スマートタップ

高精度スマートタップ

<< IEEE1888機器を使ってみる >>

GWの実例3 (BACnet)



<< IEEE1888機器を使ってみる >>

GWの実例4 (独自通信)

IEEE1888通信

ゲートウェイ

muNet通信

GE社スマートメータ
(I-210+n, I-210+c)

一般家庭: 4台設置 (4戸)
旅館: 1台設置
弟子屈町役場: 2台設置

北海道弟子屈町にて
運用中

<< IEEE1888機器を使ってみる >>

GWの実例5 (CSVファイル)

↕ IEEE1888通信

ゲートウェイ

↕ Windowsファイル共有

HISMAC

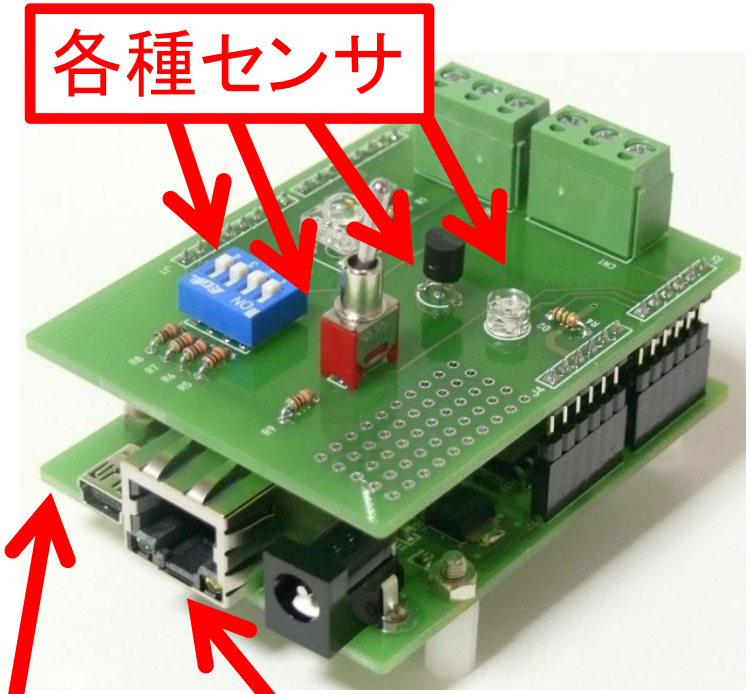
66000V 受変電設備

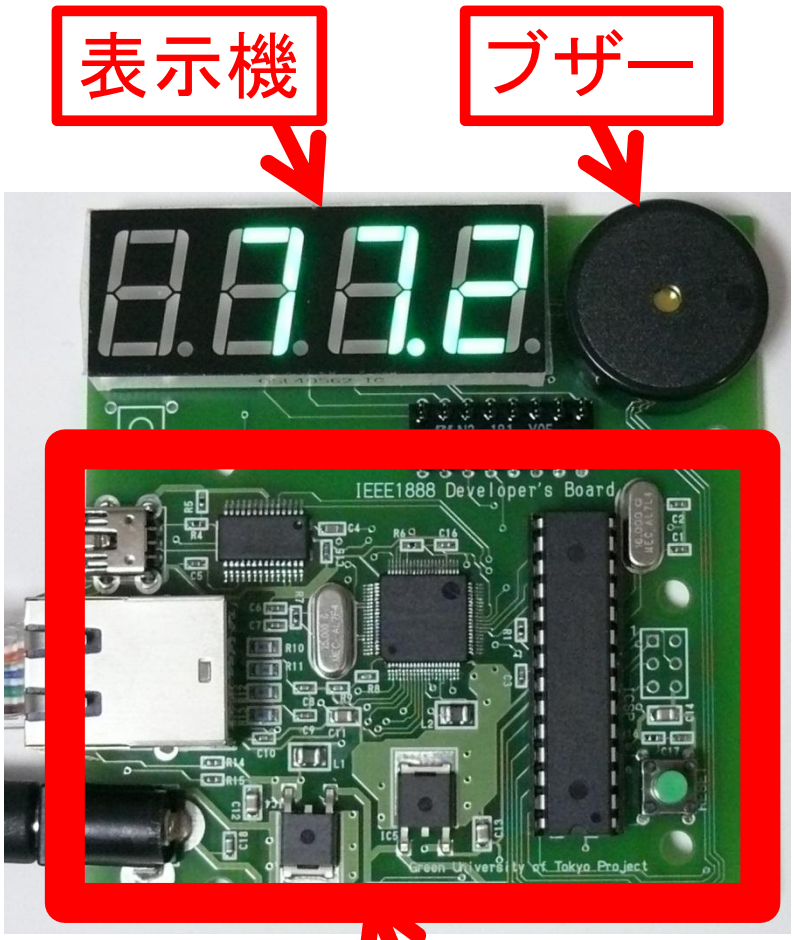
CSVファイル

東京大学本郷キャンパスにて
運用中

<< IEEE1888機器を使ってみる >>

GWの実例6 (独自回路)





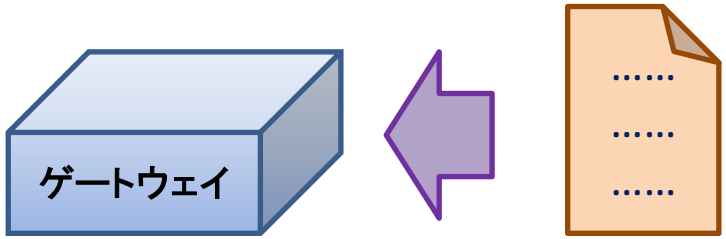
講習会ではこのタイプのゲートウェイを使います

<< IEEE1888機器を使ってみる >>

GWの設定と動作

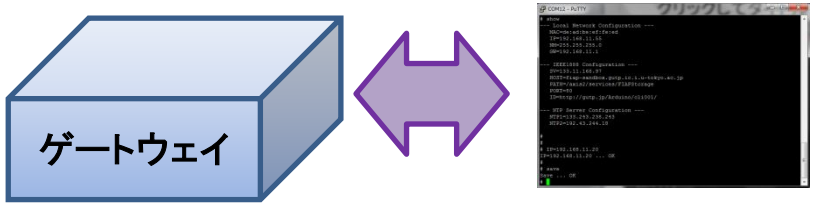
ゲートウェイを設定するには、いくつかの方法があります！
(対象とするGWによって、方法は異なります)

①



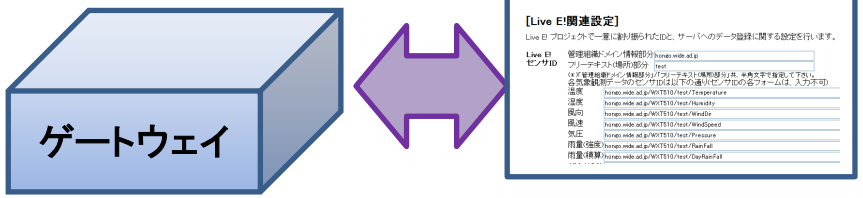
設定ファイルを(SDカードなどで)
注入する方式

②



コマンドライン・インタフェースで
インタラクティブに設定する方式

③



グラフィカル・インタフェースで
インタラクティブに設定する方式

<< IEEE1888機器を使ってみる >>

GWの設定と動作

コマンドライン・インタフェースで設定する場合の例

COM20 - PuTTY

```
Starting IEEE1888 Learning Kit Ver.1.0 ...
DHCP requesting ...  done
NTP querying ...
NTP sync. time=1338481904
> show
--- Local Network Configuration ---
   MAC=02:ad:be:ef:fe:ed
   DHCP=true
   IP=192.168.11.5
   NM=255.255.255.0
   GW=192.168.11.1
   DNS=192.168.11.1

--- IEEE1888 Configuration ---
   HOST=fiap-sandbox.gutp.ic.i.u-tokyo.ac.jp
   PATH=/axis2/services/FIAPStorage
   PORT=80
   ID=http://gutp.jp/Arduino/test-006/
   AutogenID(Temperature): http://gutp.jp/Arduino/test-006/Temperature
   AutogenID(Illuminance): http://gutp.jp/Arduino/test-006/Illuminance
   AutogenID(DIPSW): http://gutp.jp/Arduino/test-006/DIPSW
   AutogenID(TGLSW): http://gutp.jp/Arduino/test-006/TGLSW

--- NTP Server Configuration ---
   NTP1=133.243.238.243
   NTP2=192.43.244.18

>
```

② ターミナル・ソフトウェア(putty) を使い
コマンドライン・インタフェースにより、
設定する

①パソコンとUSBケーブルで接続する

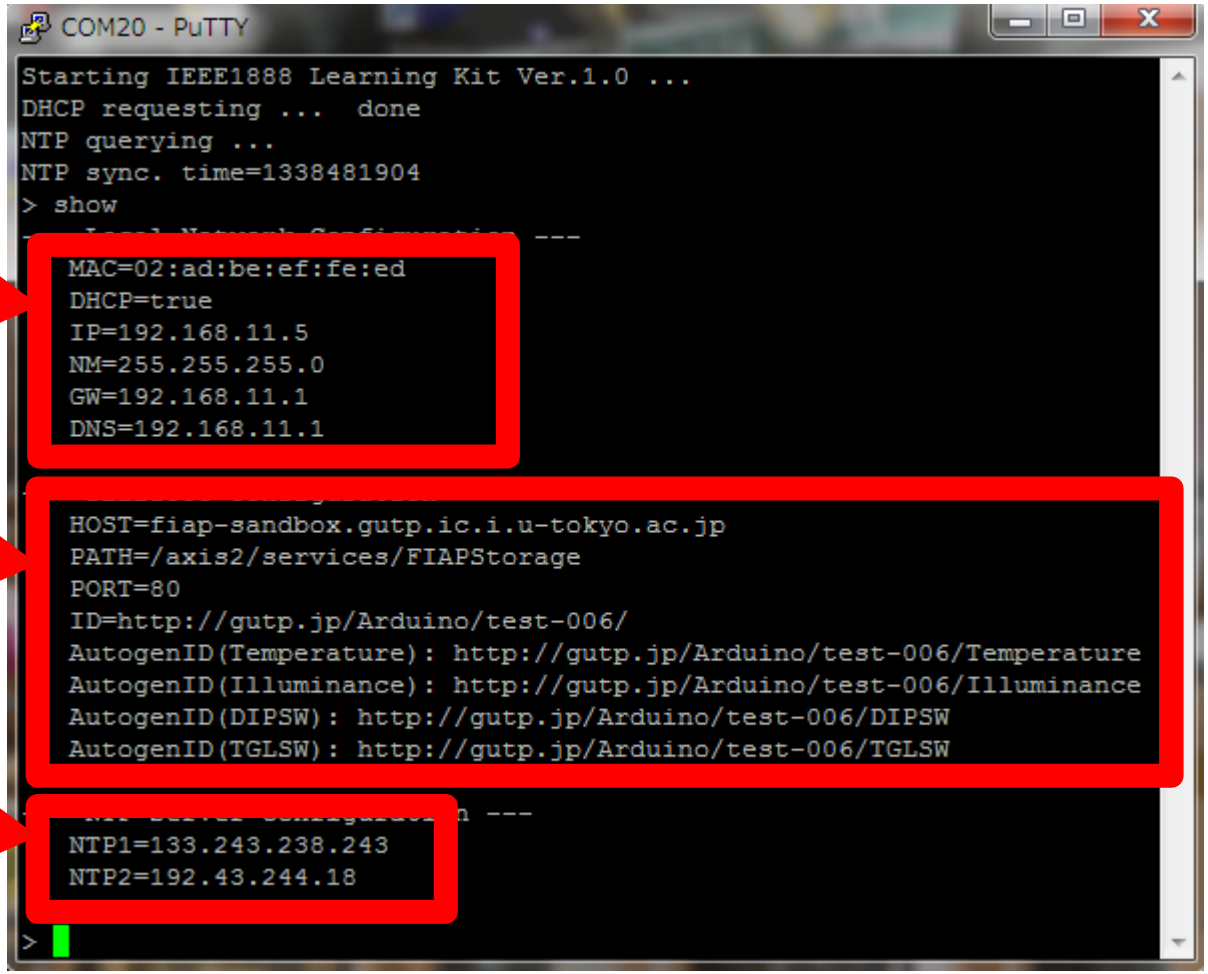
<< IEEE1888機器を使ってみる >>

GWの設定と動作

ローカルネットワーク設定
・DHCP有効無効設定
・IPアドレス、ネットマスク、
ゲートウェイ、DNS設定

IEEE1888通信設定
・サーバ設定
・ポイントID設定

NTP(時刻サーバ)設定



設定内容に応じてネットワークに接続され、観測データがサーバに送信されます。

実践: 設定の投入と動作の確認 (1/8)

※ 次のように設定してみましょう!!

DHCP: false
IPアドレス : 192.168.10.XX1
ネットマスク: 255.255.255.0
ゲートウェイアドレス: 192.168.10.254
DNSアドレス: 192.168.10.254

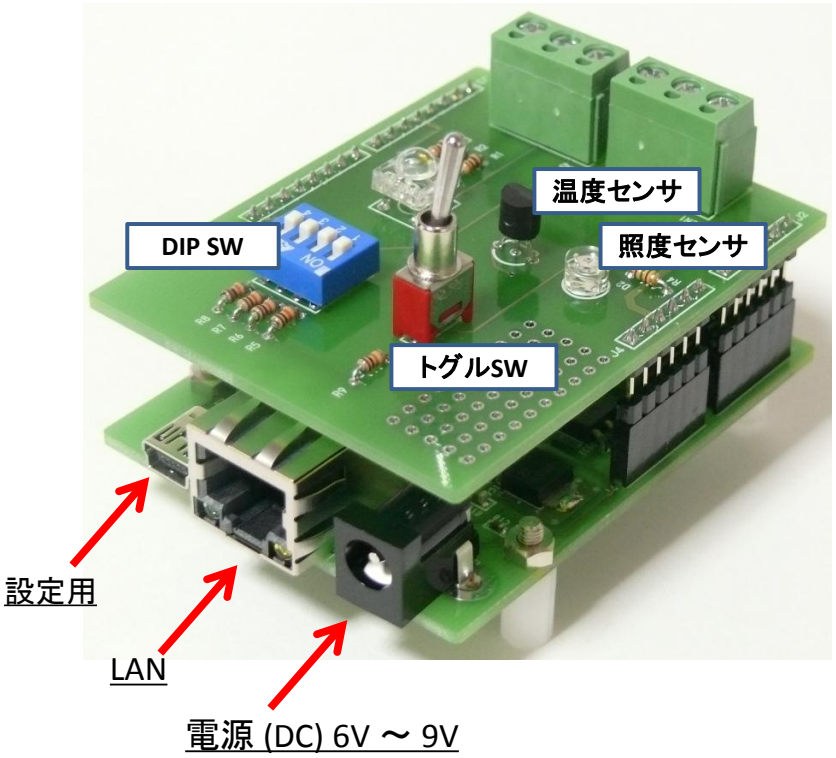
IEEE1888サーバ(ホスト名): 133.11.168.3
IEEE1888サーバ(パス名): /axis2/services/FIAPStorage
IEEE1888サーバ(TCP ポート番号): 80
IEEE1888ポイントID(プレフィックス): http://gutp.jp/interopXX/

NTPサーバアドレス1: 133.243.238.243
NTPサーバアドレス2: 192.43.244.18

(*) XXには各グループ番号を設定してください

実践: 設定の投入と動作の確認 (2/8)

☆ゲートウェイの仕様を確認しよう！！



IEEE1888 学習用ゲートウェイ・キット

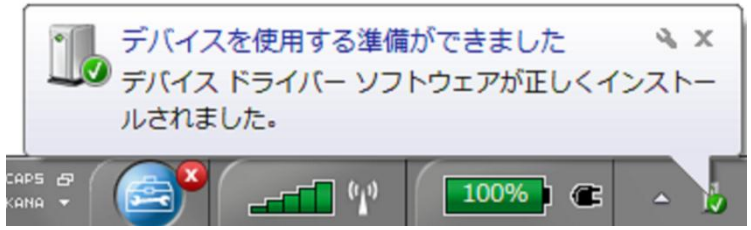
学習用ゲートウェイの仕様

項目	仕様
温度	0 ～50 ℃ (小数点1桁) (学習用のため、値は正確ではありません)
照度	0 ～ 1000 ルクス (整数) (学習用のため、値は正確ではありません)
DIPスイッチ	0 ～15 (整数)
トグルスイッチ	ON または OFF
IEEE1888	WRITEクライアント (送信先は1ヵ所設定可) ポイント数:4 (各センサ毎)
送信頻度	毎分 0 秒 (固定)
インターネット	DNS対応、DHCP対応、NTP対応、IPv4のみ
Ethernet	Auto MDIX (10M/100M)
設定方法	USBシリアル(コマンドライン)
電源	6V ～ 9V (300mA)

実践: 設定の投入と動作の確認 (3/8)

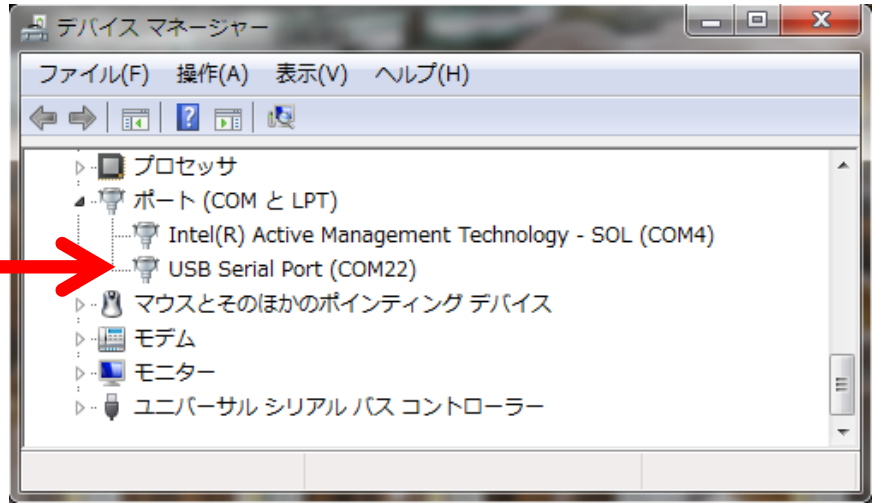
☆設定の準備をしよう！！(物理的な接続編)

① パソコンにUSBで接続します



② パソコンにデバイスドライバがインストールされます(自動)

③ デバイスマネージャを開き、COMポート番号を調べます。
（この例ではCOM22）

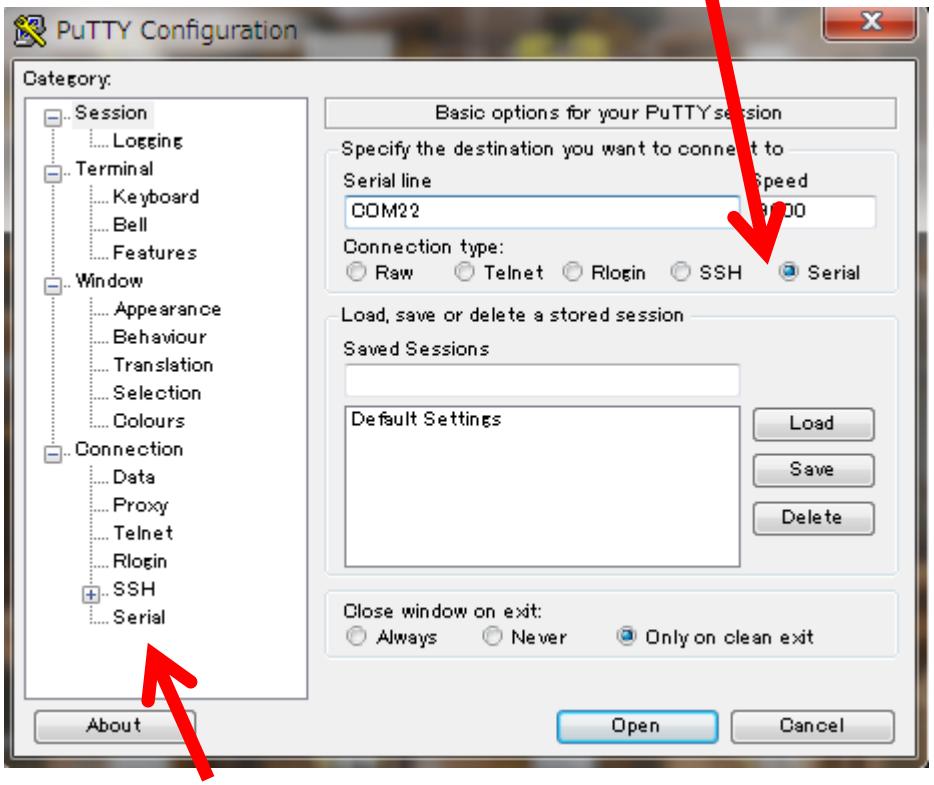


(*) デバイスマネージャは、
(コントロールパネル→ システム)
から開くことができる

実践: 設定の投入と動作の確認 (4/8)

☆設定の準備をしよう！！(ソフトウェア的な接続編1)

① シリアル通信モード(Serial)を選択



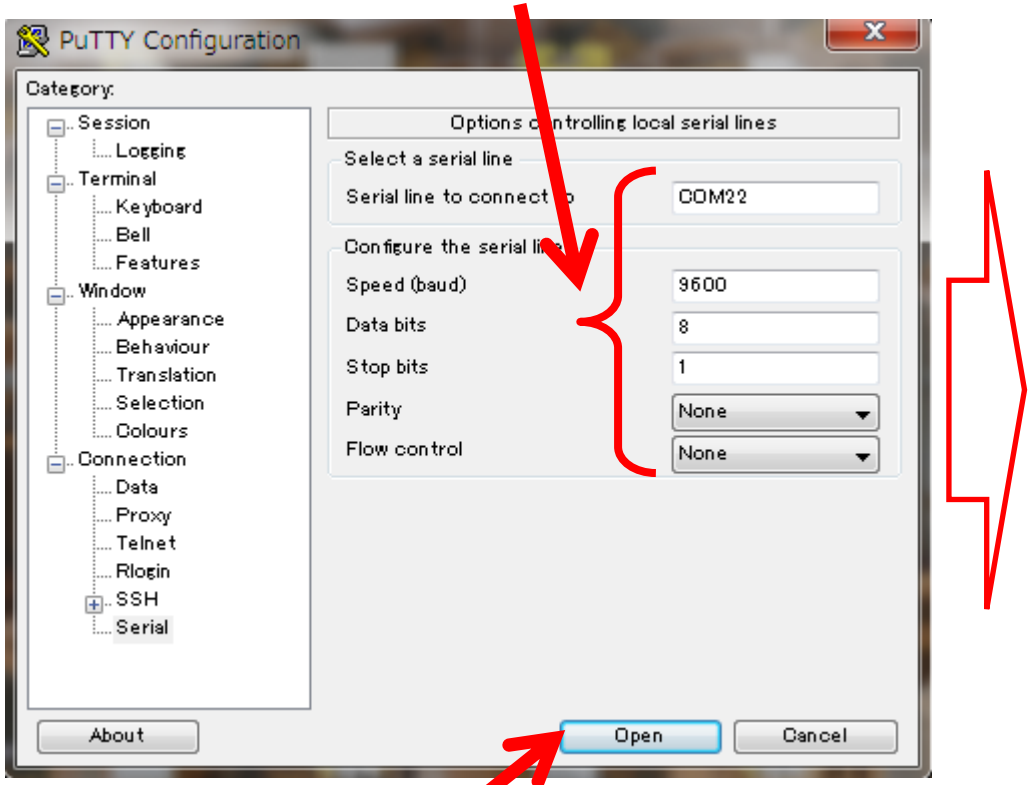
② Serial詳細設定を選択

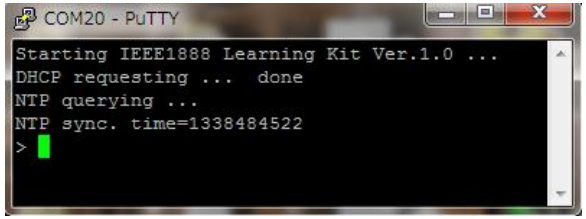
次スライドへ

実践: 設定の投入と動作の確認 (5/8)

☆設定の準備をしよう！！(ソフトウェア的な接続編2)

前スライドより

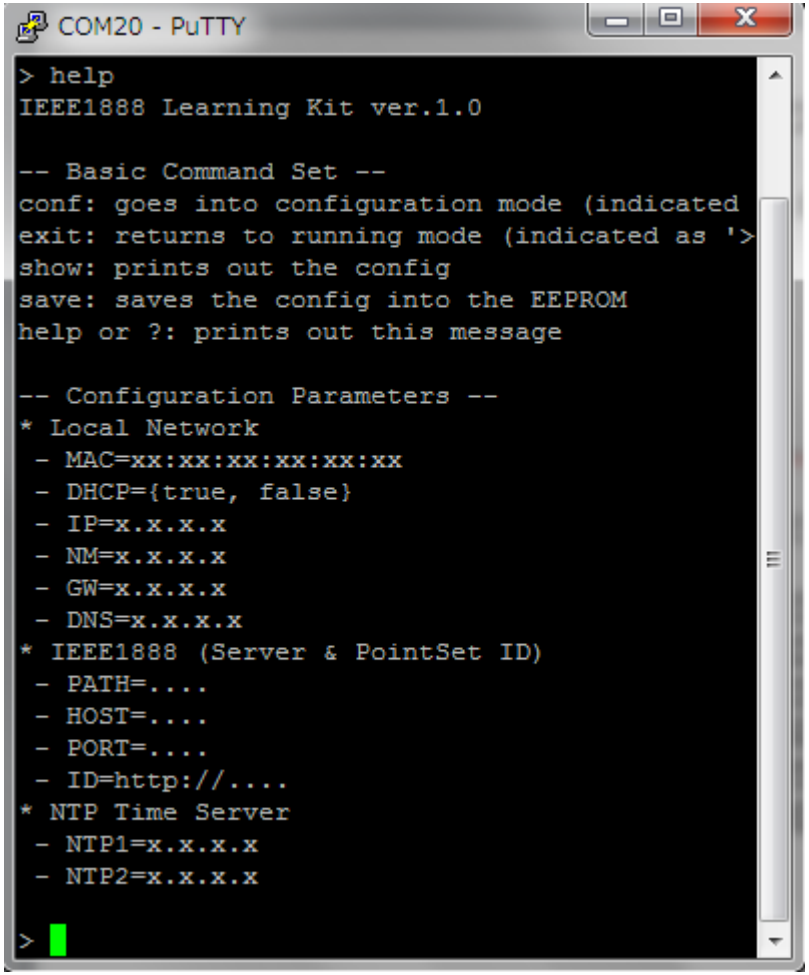




⑤ 接続完了

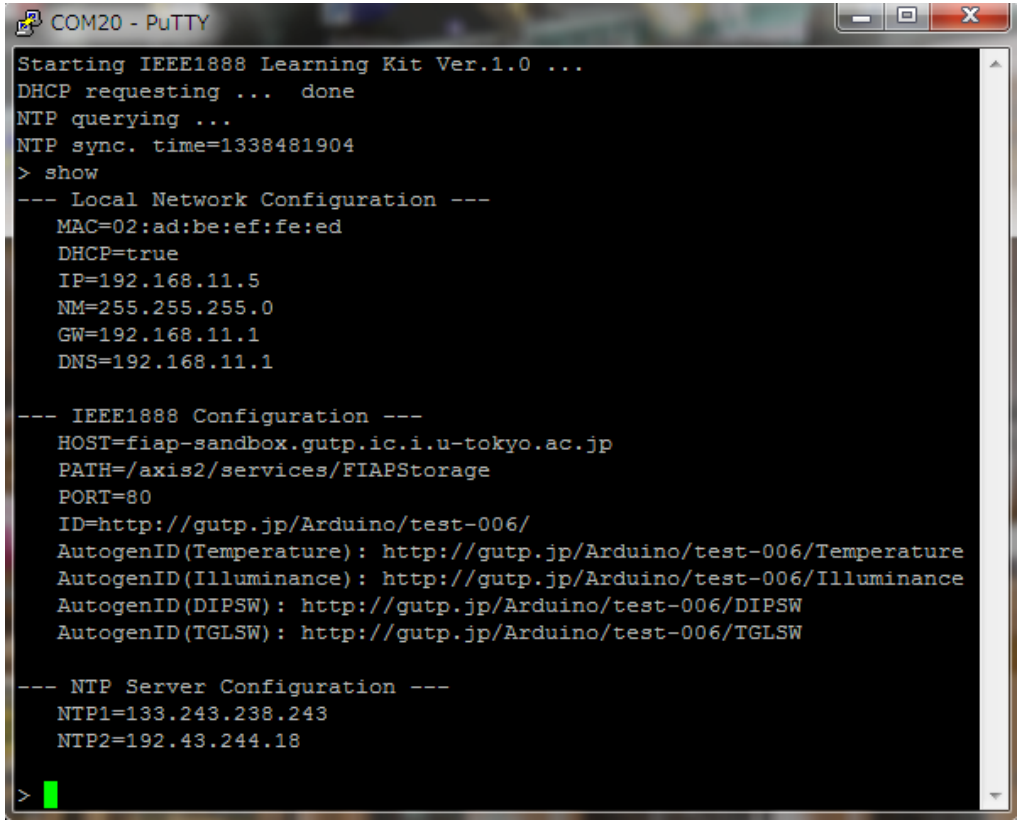
実践: 設定の投入と動作の確認 (6/8)

helpコマンドを実行してみる！！



コマンド＆設定パラメーター一覧が表示されます。

show コマンドを実行してみる！！

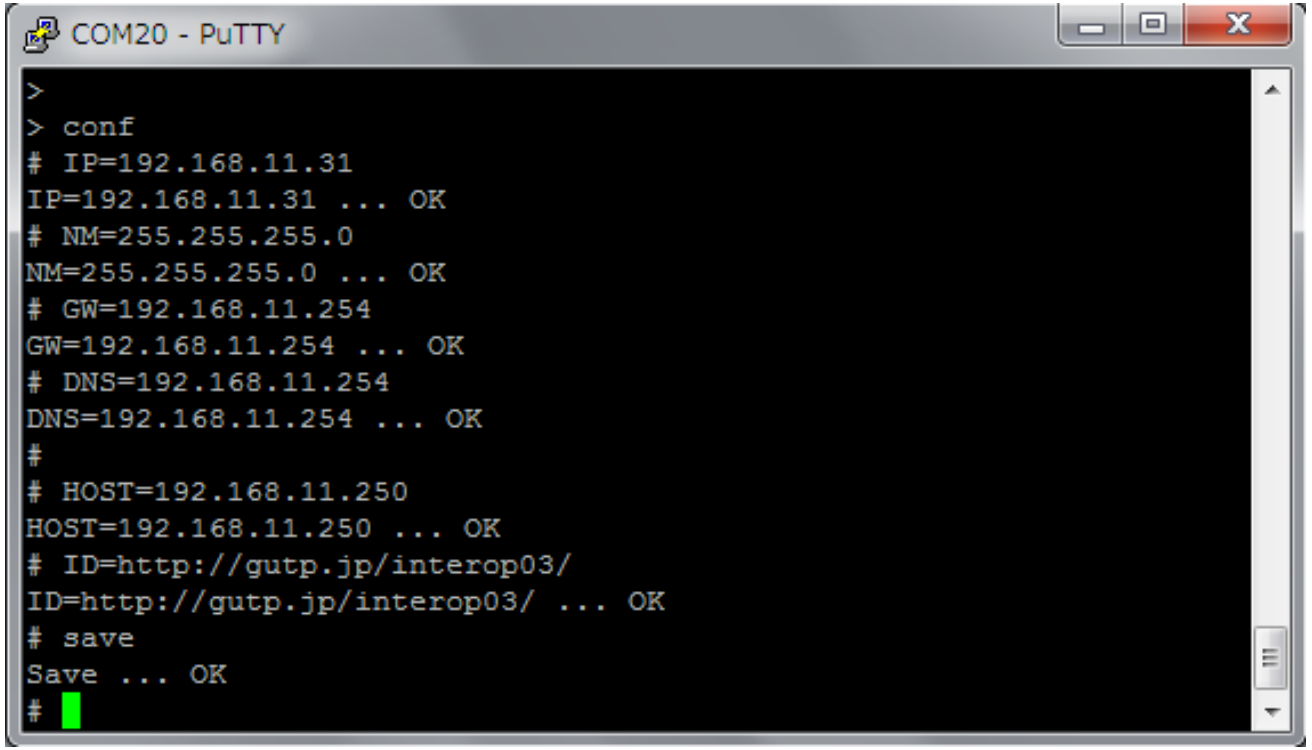


現在の設定内容が表示されます。

実践: 設定の投入と動作の確認 (7/8)

■設定内容を変更するには

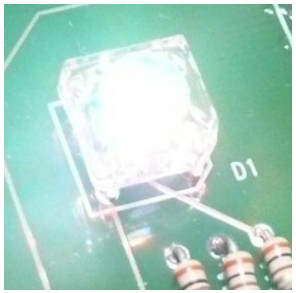
1. conf コマンドを実行 (送信動作が止まり、設定モードになる)
2. パラメータを設定する (必要に応じてshowコマンドで設定内容を確認する)
3. Saveコマンドを実行 (これにより設定内容を保存する)
4. 再起動



一連の設定変更の流れ

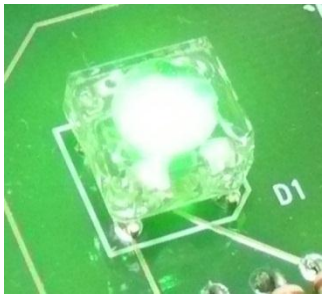
実践: 設定の投入と動作の確認 (8/8)

動作確認



白:NTP通信中

青:データ送信中



緑:データ送信成功

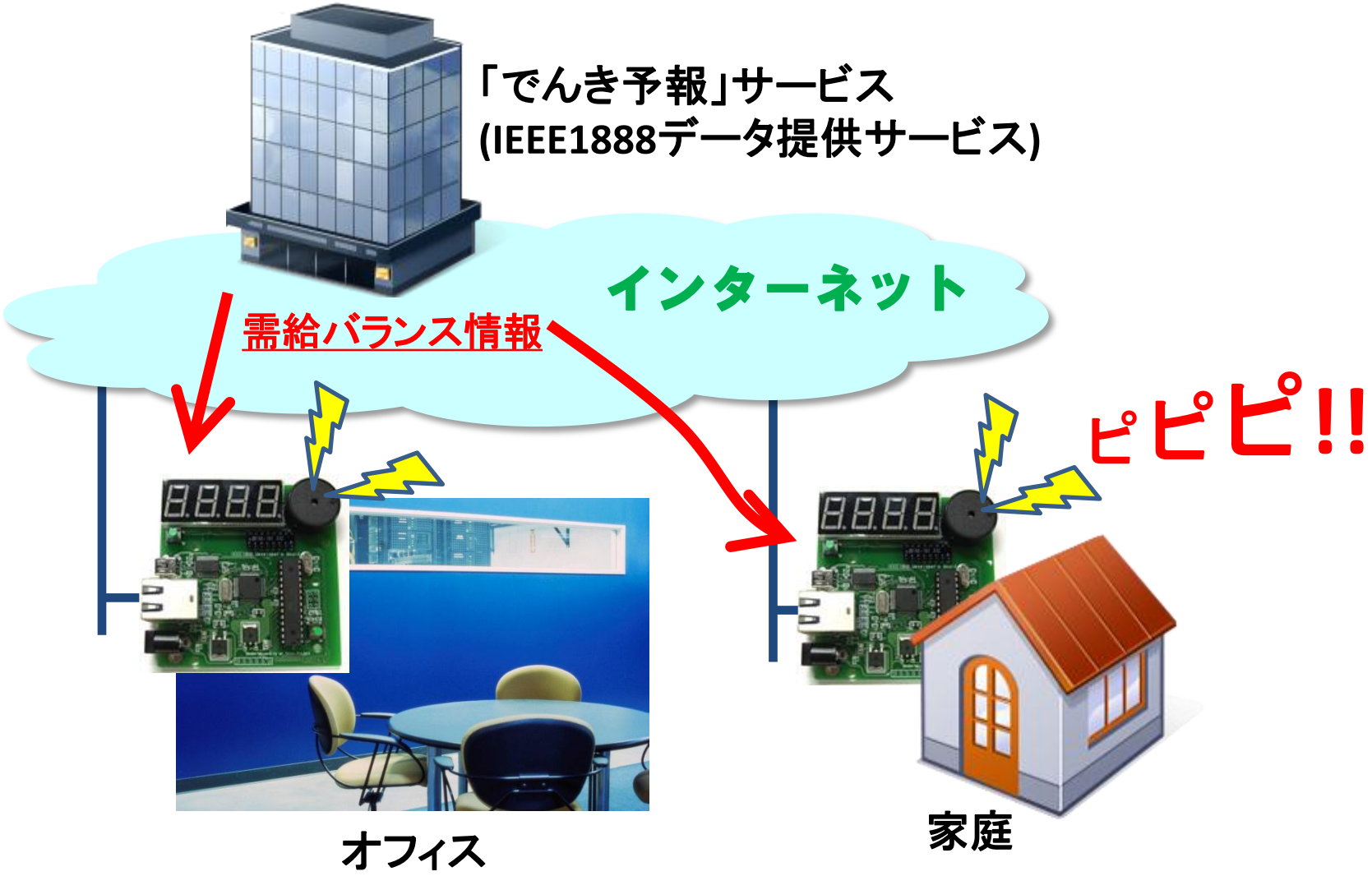
紫色:サーバへの接続失敗
（TCP接続失敗）
黄色:HTTP通信エラー
水色:IEEE1888通信エラー
赤色:その他のエラー

サーバ側のデータ確認

http://133.11.168.3/ をブラウザで確認する

Point ID	Time	Value
http://gutp.jp/interop00/DIPSW	2012-06-03T19:46:00.000+09:00	0
http://gutp.jp/interop00/Illuminance	2012-06-03T19:46:00.000+09:00	265
http://gutp.jp/interop00/TGLSW	2012-06-03T19:46:00.000+09:00	ON
http://gutp.jp/interop00/Temperature	2012-06-03T19:46:00.000+09:00	30.2

デマンド警報装置についてもやってみよう



実習で使用したGWとアドバンストなGW

- 実習で使用したGW
  - WRITEクライアント機能
  - FETCHクライアント機能

} 対向にIEEE1888サーバの存在が前提

- アドバンストなGWは
  - 上記機能に加え、
    - WRITEサーバ機能
    - FETCHサーバ機能
    - TRAPサーバ機能

    を実装している

本日(15日)のスケジュール

10:00　IEEE1888とは何か？（全体像）

11:00　IEEE1888機器を使ってみる

12:00　昼食

13:00　IEEE1888ソフトウェア開発キット(SDK)の概要

13:30　SDKを使ったFETCH手順のプログラミング

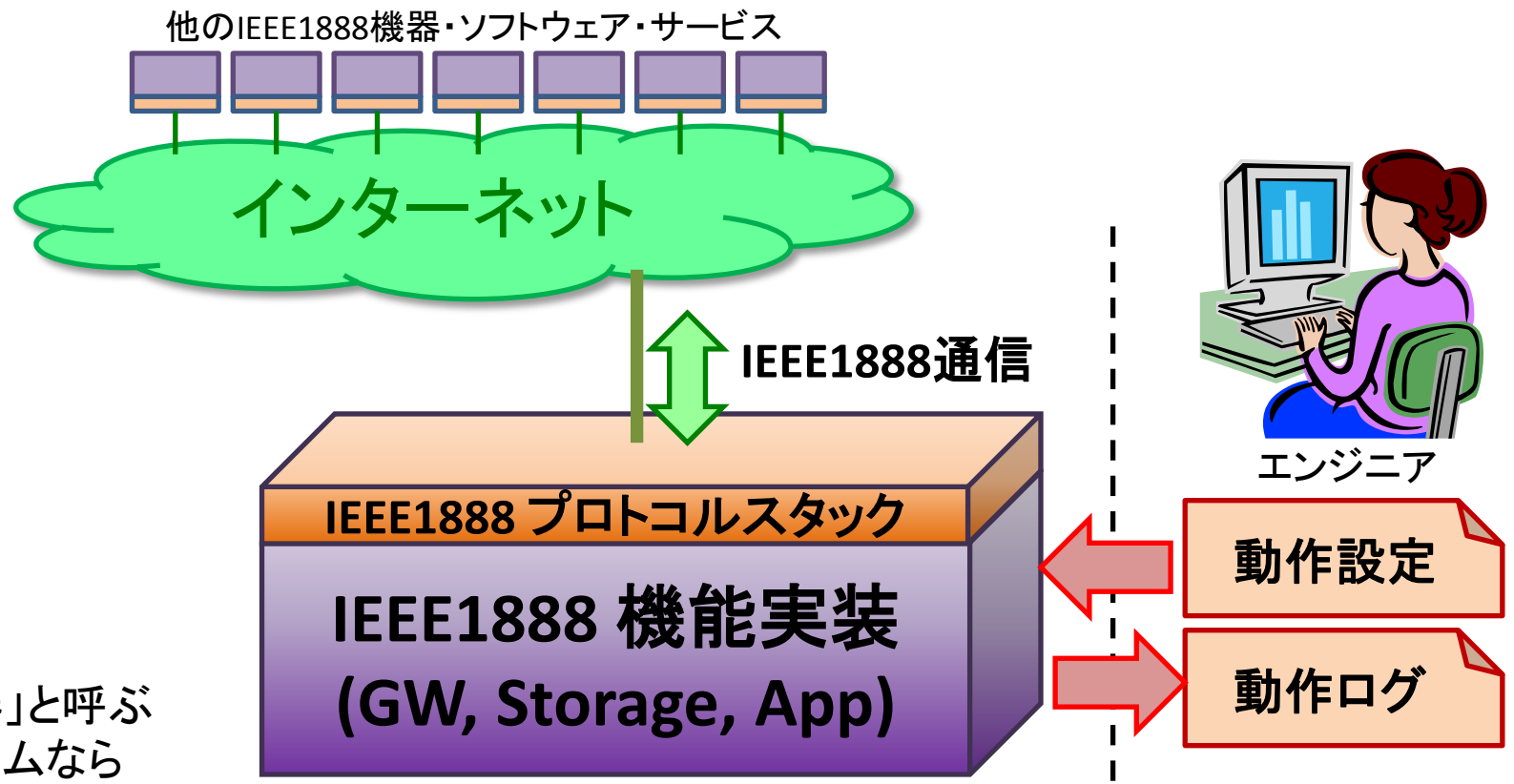
15:00　休憩

15:30　SDKを使ったWRITE手順のプログラミング

16:30　C言語版/Arduino版についての紹介

17:00　解散

<< IEEE1888 SDK を使い IEEE1888機器を開発する >>
機器・ソフトウェア・サービスの一般的なアーキテクチャ



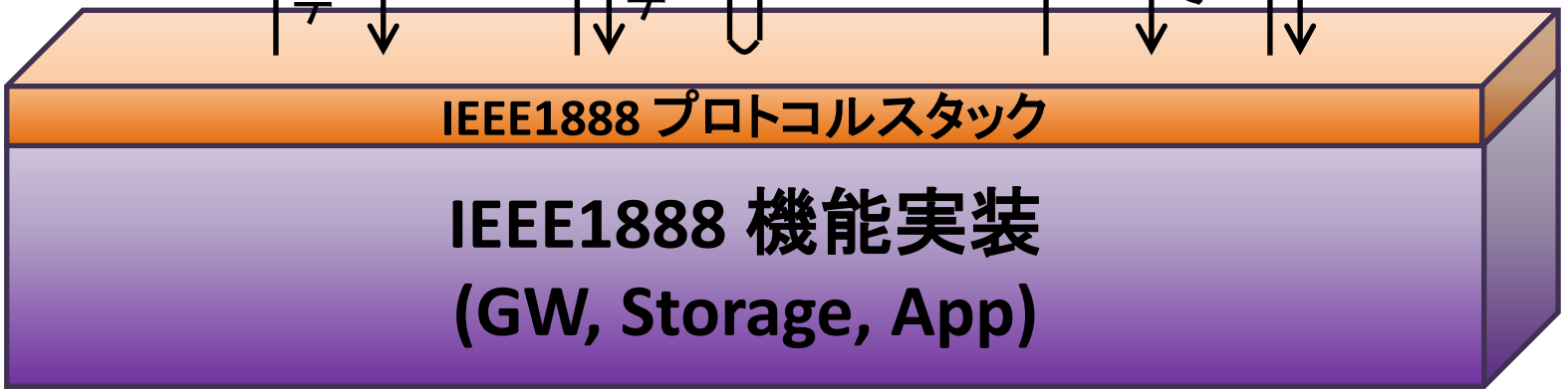
・箱なら
➔「機器」と呼ぶ
・プログラムなら
➔「ソフトウェア」と呼ぶ
・ネットワークの向こうで動くなら
➔「サービス」と呼ぶ

<< IEEE1888 SDK を使い IEEE1888機器を開発する >>

実装する通信機能を決める

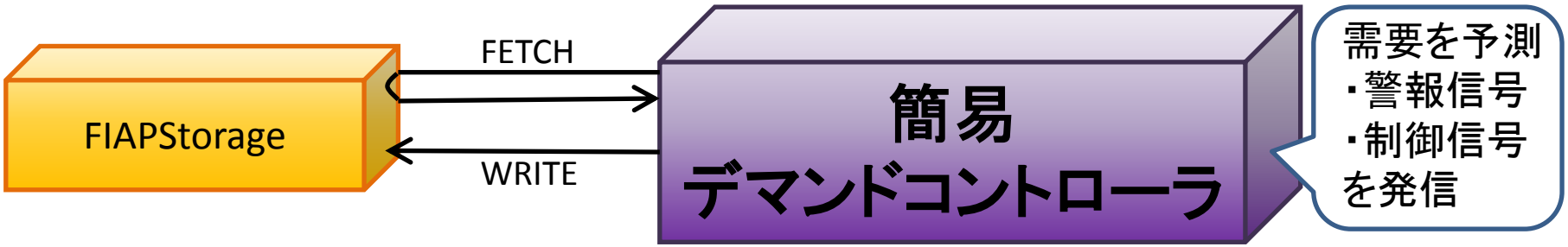
例えば、午前中のワークショップで使った装置は、

	WRITEクライアント	WRITEサーバ	FETCHクライアント	FETCHサーバ	TRAPリクエスト	TRAPコールバック	TRAPサーバ
温度・照度センサ	○	×	×	×	×	×	×
簡易警報機	×	×	○	×	×	×	×
Storageサーバ	×	○	×	○	×	×	×
見える化APP	×	×	○	×	×	×	×



<< IEEE1888 SDK を使い IEEE1888機器を開発する >>
実装機能＆設定ファイルの設計 (1/3)

IEEE1888による簡易デマンド・コントローラ（ソフトウェア）の場合



■動作の概要

FIAPStorageの使用を前提とする(本ソフトウェアは、FETCHクライアント、WRITEクライアントとして動作)。
電力データ(有効電力量)はFIAPStorageに集められているものを使用する。目標デマンド値をFIAPStorageからも読めるようにする（静的設定も可能にする）。
このソフトウェアにより生成される、警報信号、負荷遮断信号、ステータス情報はすべてFIAPStorageに保存する。

30分毎に、phase0 → phase1 → phase2と動作状態が遷移する。30分が経過すると、またphase0から始まる。
phase0: すべての警報は解除され、負荷への電力供給が行われる。
phase1: 5分ごとにデマンドを予測する。その予測結果に応じて、レベル1、レベル2、レベル3の警報を発令。レベル3の場合には、負荷への電力を遮断する。
phase2: 1分ごとにデマンドを予測する。その予測結果に応じて、レベル1、レベル2、レベル3の警報を発令。レベル3の場合には、負荷への電力を遮断する。

デマンド予測方式は、メータの変分の延長上(30分の時点)から算出する。警報状態への移行・警報状態からの解除には、ヒステリシス特性を持たせる。
30分が経過した時点で、レベル1発令時間、レベル2発令時間、レベル3発令時間、負荷遮断時間を、ステータス情報として記録する。

<< IEEE1888 SDK を使い IEEE1888機器を開発する >>

実装機能＆設定ファイルの設計 (2/3)

IEEE1888による簡易デマンド・コントローラ（ソフトウェア）の場合 (つづき)

■設定ファイルのフォーマット

```
<fiapSimpleDemacon fiapStorageURL="..."
 xmlns="http://gutp.jp/fiapSimpleDemacon/2012/06/">

 <meter id="..." clearAt="..." />
 <demand phase1="10" phase2="25"
       level1="..." level1_id="..." level1_hys="1"
       level2="..." level2_id="..." level2_hys="1"
       level3="..." level3_id="..." level3_hys="1" />

 <warningLevel1 id="..." on="true" off="false" />
 <warningLevel2 id="..." on="true" off="false" />
 <warningLevel3 id="..." on="true" off="false" />
 <loadSwitch id="..." on="true" off="false" />

 <warningDurationLevel1 id="..."/>
 <warningDurationLevel2 id="..."/>
 <warningDurationLevel3 id="..."/>
 <loadSwitchOffDuration id="..."/>

</fiapSimpleDemacon>
```

FIAPStorageの通信先設定

電力メータ(情報源)の定義

デマンド予測と警報レベル設定

警報発令時の信号生成ルール

動作統計の保存ルール

<< IEEE1888 SDK を使い IEEE1888機器を開発する >>
実装機能＆設定ファイルの設計 (3/3)
IEEE1888による簡易デマンド・コントローラ（ソフトウェア）の場合 （つづき）

■設定ファイルでの各項目の説明(詳細)

* fiapSimpleDamecon (デマンドコントローラの設定)
 fiapStorageURL(URL): IEEE1888通信(FETCH, WRITE)の対象とするStorageサーバのSOAP-EPR。

* meter (電力メータの設定)
 id(URI): 電力メータ(積算有効電力量)を表すポイントID
 clearAt(float): 電力メータの周期を指定する (99999 -> 0 となる電力メータの場合、100000を指定)

* demand (デマンド量に関する設定)
 phase1(int): phase0 -> phase1に移行する時刻 (0分-30分の間の整数＆phase1<=phase2)
 phase2(int): phase2に移行する時刻 (0分-30分の間の整数＆phase1<=phase2)
 level1[level1_idが指定されたら省略可](float): 警報レベル1の閾値(デマンド量)
 level1_id[省略可](float): level1の値をFIAPStorageから取得する場合のポイントID (level1が指定されていればそちらの値を優先)
 level1_hys[省略可(省略時0)](float: 0-1): 警報レベル1のヒステリシス特性の設定。例えば、level1="100"でlevel1_hys="0.01"であれば、デマンド予測が、100を越えたら警報レベル1に突入。警報レベル1を解除するには、デマンド予測が、100 - 100*0.01 = 99 以下にならなければならない。

 level2*, level3* についても同様。

* warningLevel1 (警報レベル1の出力に関する設定)
 id(URI): 警報レベル1を示すポイントID
 on(文字列)[省略可(省略時true)]: 警報レベル1の発令を意味する文字列
 off(文字列)[省略可(省略時false)]: 警報レベル1を発令していないことを意味する文字列
 (上位警報が発令中は、警報レベル1の発令も包含する)

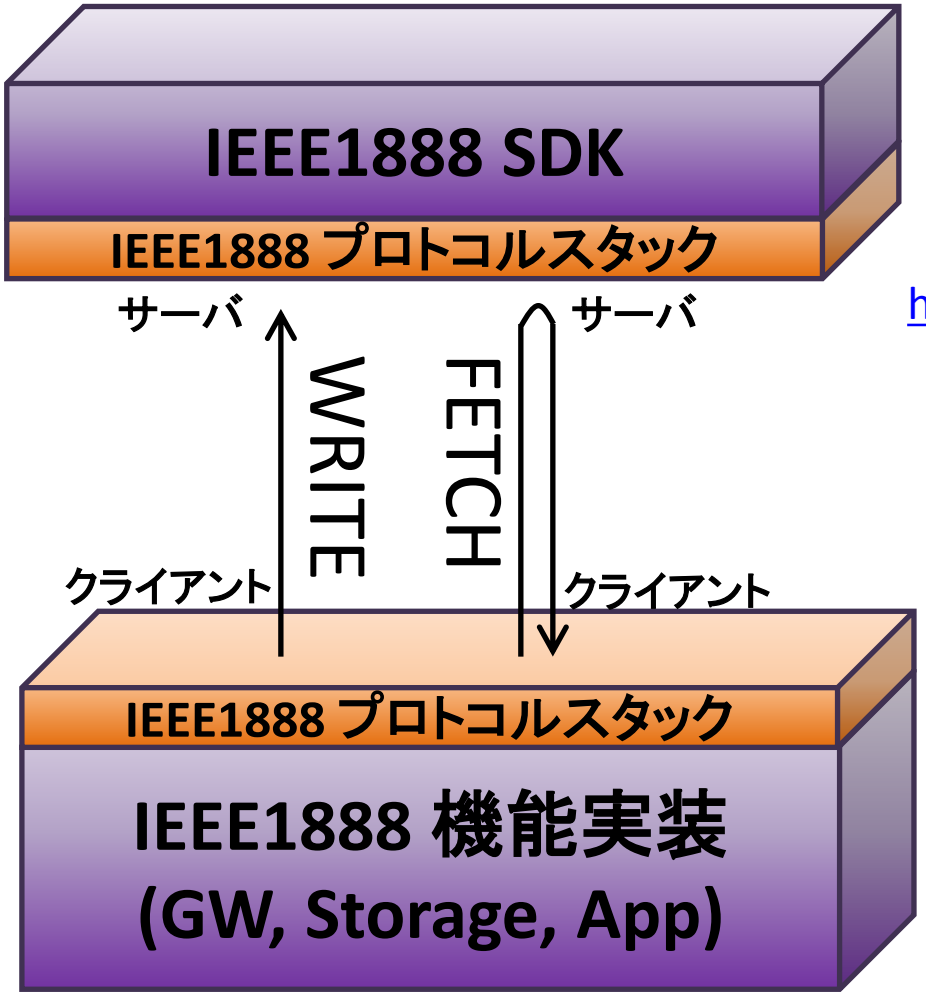
* warningLevel2, warningLevel3も同様

* loadSwitch (負荷のON/OFF制御に関する設定)
 id(URI): 負荷のON/OFF制御を示すポイントID
 on(文字列)[省略可(省略時true)]: 負荷をONにすることを意味する文字列
 off(文字列)[省略可(省略時false)]: 負荷をOFFにすることを意味する文字列

* warningDurationLevel1 (警報レベル1の期間を記録する)
 id(URI): 警報レベル1が発令されていた時間(秒)を記録するポイントID
 30分ごとに記録される(時刻は、時限の開始時刻とする)

* warningDurationLevel2, warningDurationLevel3, loadSwitchOffDuration についても、同様。

IEEE1888ソフトウェア開発キット
SDK: Software Development Kit



IEEE1888 SDK (2011年版)は
・WRITEサーバ機能
・FETCHサーバ機能
を提供する

http://gutp.jp/fiap/kit.htmlから入手可能
VMware Player仮想マシンでの配布

IEEE1888 SDKを使うことで
・WRITEクライアント機能
・FETCHクライアント機能
を持つ、IEEE1888部品を開発することができる。

IEEE1888 SDK は VMware Player 上で実行

ダブルクリック

または

「再生」をクリック

インストール方法はSDKに付属の「IEEE1888SDKインストールマニュアル.pdf」を参照

IEEE1888 SDK の稼働画面

ポイント一覧（データ確認）
Storageに蓄積されているデータを表示

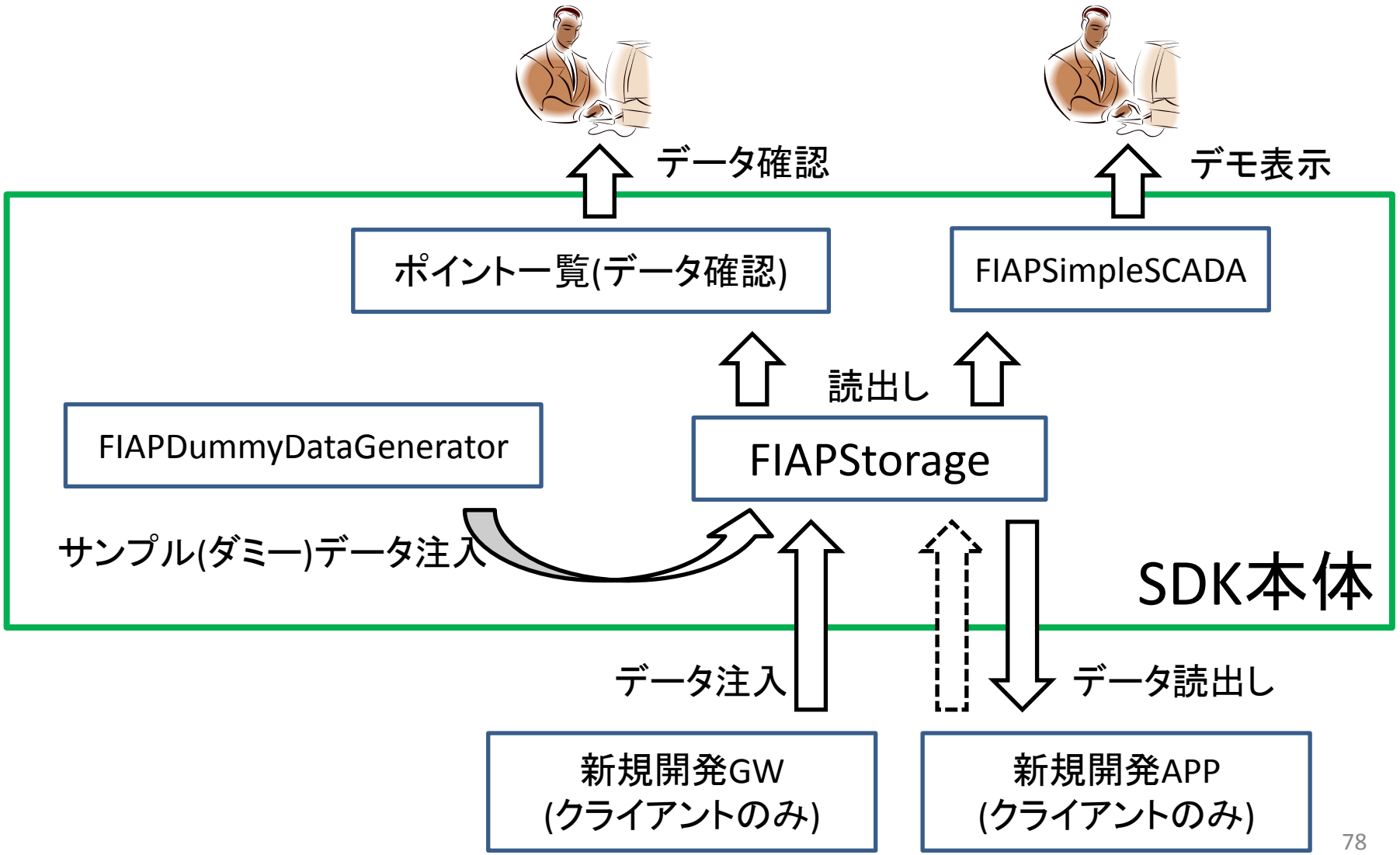
FIAPSimpleSCADA
GWからの収集データを整理して表示
サンプルアプリでもある

端末
IPアドレスの確認等

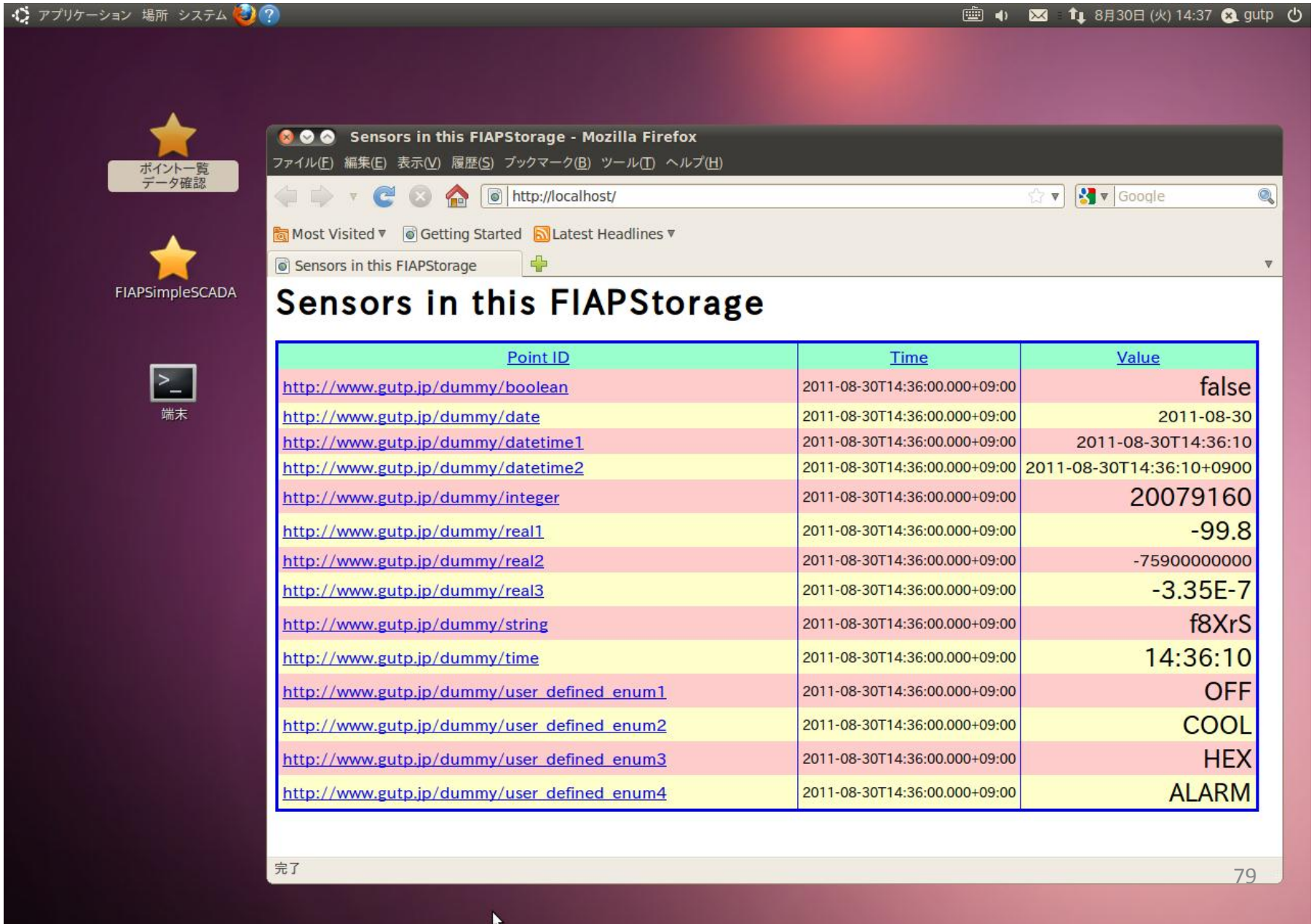
PHP5のサンプルコード

詳細はSDKに付属の「IEEE1888SDKスタートアップマニュアル.pdf」を参照

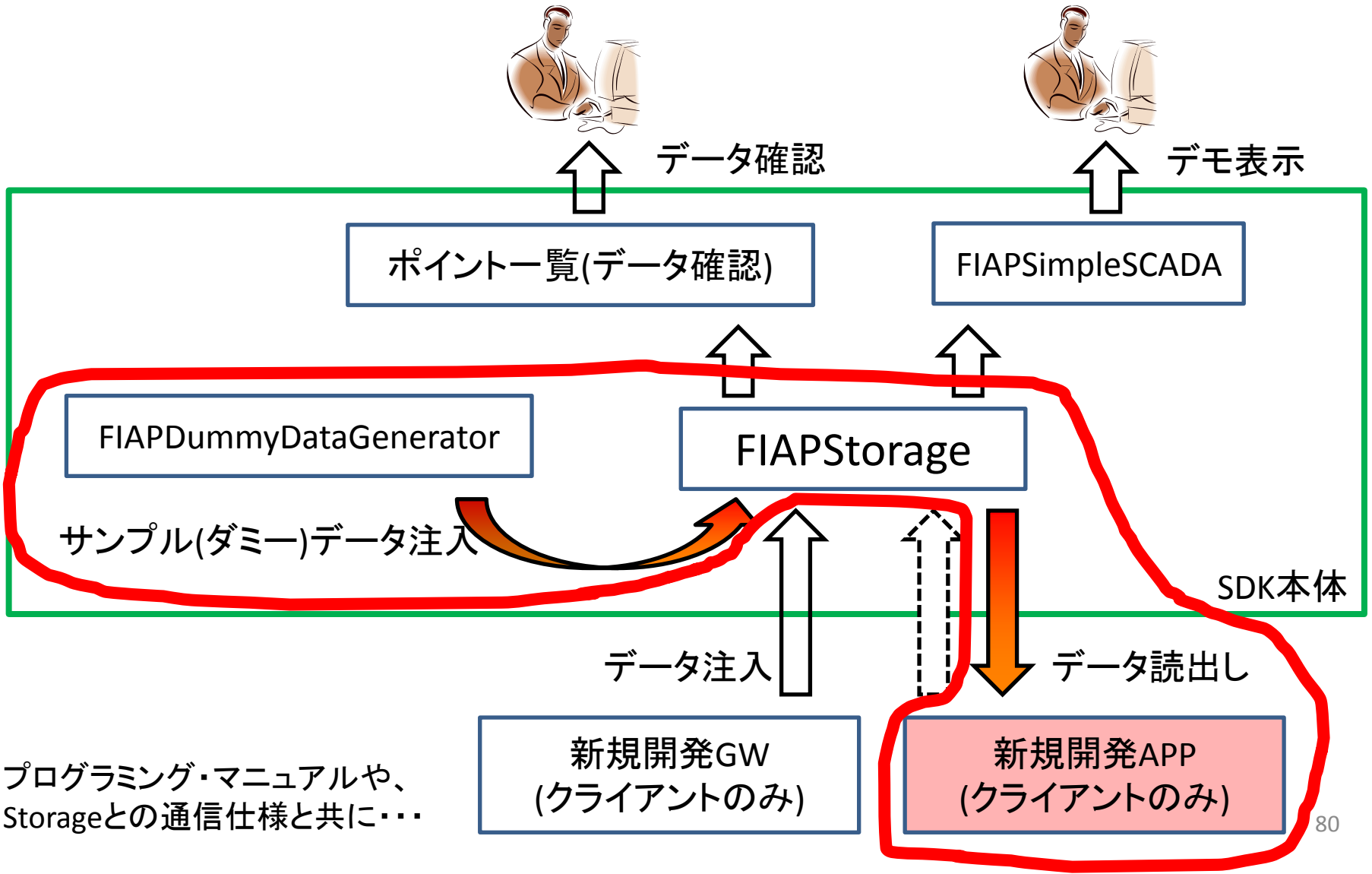
SDK 構成要素間の関係



ポイントー覧(データ確認)画面

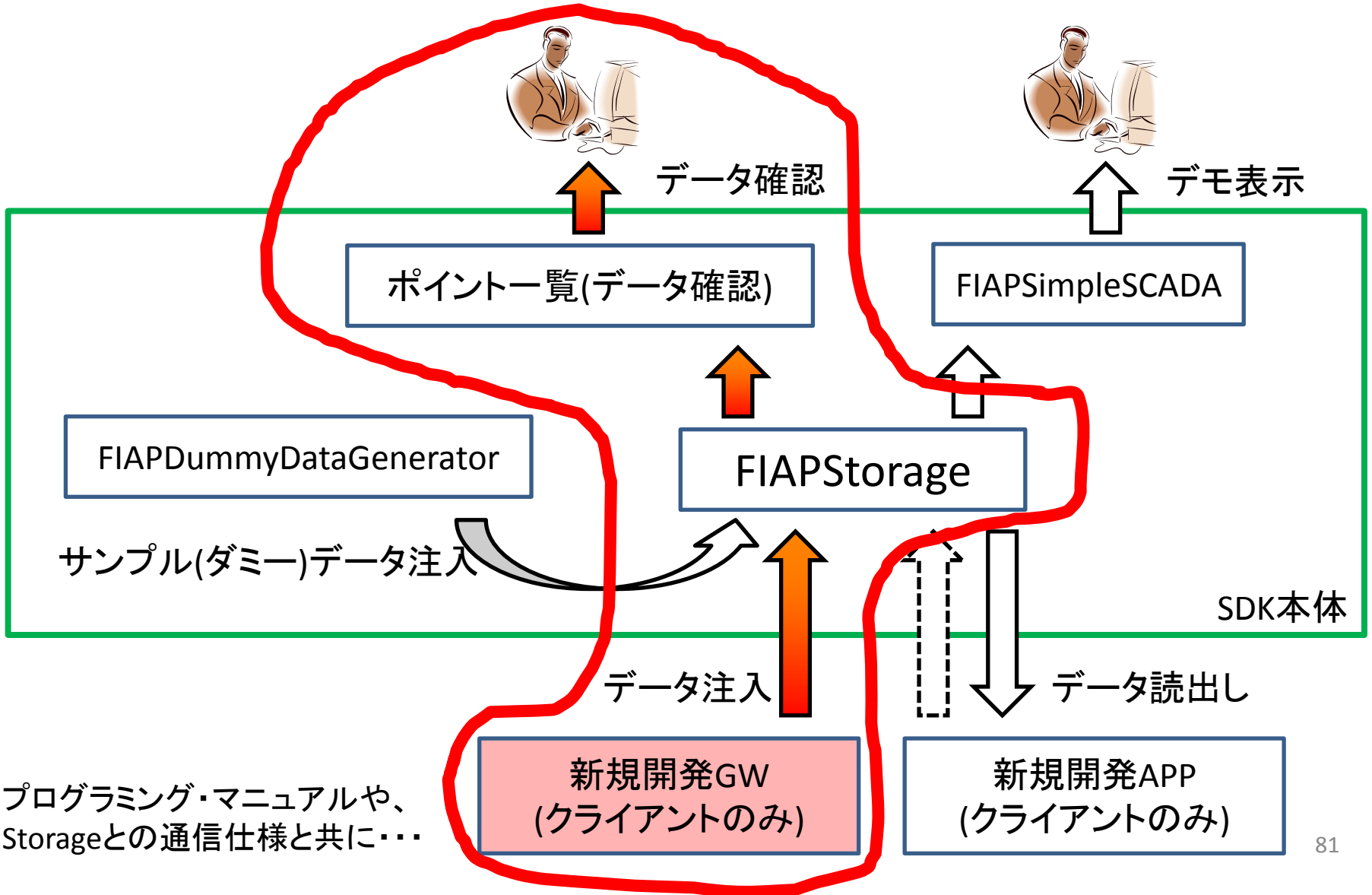


FETCHクライアントを作成する場合に使用する構成要素



プログラミング・マニュアルや、
Storageとの通信仕様と共に・・・

WRITEクライアントを作成する場合に使用する構成要素



プログラミング・マニュアルや、
Storageとの通信仕様と共に・・・

SDKのファイル構成

IEEE1888SDK-20120617

+ --- Documents

| +--- IEEE1888SDKインストールマニュアル.pdf

| +--- IEEE1888SDKスタートアップマニュアル.pdf

| +--- IEEE1888プログラミングスタートアップマニュアル.pdf

| +--- IEEE1888 Storageとの通信仕様.pdf

| +--- FIAPDummyDataGenerator仕様書兼インストールマニュアル.pdf

| +--- FIAPSimpleSCADA仕様書兼インストールマニュアル.pdf

|

+--- Virtual Machines

+--- IEEE1888SDK --- VMware Player用 仮想マシン (フォルダ)

SDKのインストール＆セットアップ

インストールマニュアルおよびセットアップマニュアルに従い作業を進める

IEEE1888 SDK インストールマニュアル

東大グリーン ICT プロジェクト
プロトコル標準化 WG
最終更新: 2011 年 9 月 4 日

概要

本書は、IEEE1888 SDK のコンピュータ環境を整備するためのインストールマニュアルを記載しています。IEEE1888 SDK は VMware Player のイメージファイルとして提供されています。これを各自の PC にダウンロードし、本書に記載された手順で設定することで、利用できるようになります。本書は、第 1 章に VMware Player のインストール手順、第 2 章に IEEE1888 SDK のインストール手順を記載しています。VMware Player の環境をすでにお持ちであれば、第 1 章はスキップしてください。

1． VMware Player のインストール

VMware Player をすでにインストール済みであれば、この章はスキップしてください。

1．1．VMware Player のダウンロード

VMware のページ、
http://downloads.vmware.com/jp/d/info/desktop_downloads/vmware_player/3_0
から、VMware Player の「ダウンロード」をクリックします(図 1)。

IEEE1888 SDK スタートアップマニュアル

東大グリーン ICT プロジェクト
プロトコル標準化 WG
最終更新: 2011 年 9 月 4 日

概要

本書は、IEEE1888 SDK の構成と基本的な手順を記載しています。SDK は、IEEE1888 の GW や APP の開発を支援するソフトウェア開発キットであり、それらの開発工程を短縮するのに役立ちます。また、本 SDK を使うことで相互接続性を検証しながら GW や APP を開発できるため、より確実に開発を成功に導くことができます。

SDK は、VMware Player による仮想マシンおよび、本書を含めたドキュメントで構成されており、Web 上で公開されています。

配布 URL　http://ieee1888.gutp.ic.i.u-tokyo.ac.jp/dist/

(*) 9 月現在、ID&パスワードによるログイン制限がされています。パスワードは、GUTP 全体の ML にて公開します。

(*) VMware Player による仮想マシンのインストールについては、「IEEE1888 SDK インストールマニュアル」をご参照ください。

通信データ構造の解説

資料

IEEE1888 Storage との通信仕様

東大グリーン ICT プロジェクト(GUTP)
プロトコル標準化 WG
最終更新：2011-09-04

第 1 章 概要

本仕様書は、開発キットに含まれる IEEE1888 Storage サーバ(FIAPStorage とも呼ばれる)との通信仕様を記載している。基本的な通信プロトコルは、IEEE1888 の規定に従うが、本仕様書では、東京大学(以下、東大)が開発した IEEE1888 対応 Storage の通信仕様についてのみ記載してある。東大版の Storage は、IEEE1888 で規定される FETCH 手順、WRITE 手順、TRAP 手順のうち、FETCH および WRITE のみを実装している。

第 2 章 サーバとの通信 URL

IEEE1888 開発 SDK は VMware Player によって提供される。この仮想マシンを立ち上げると、次の場所に WSDL や SOAP 通信アドレスが用意される。

資料: IEEE1888 Storageとの通信仕様.pdf

3.1. IEEE1888 サービスのメソッド

IEEE1888 サーバは、2 種類のメソッドを持ち、それぞれ、次のように定義される。東大版 Storage からデータを読み出したり、データを書き込んだりするためには、これらのメソッドを利用する(それぞれ 3.3 および 3.4 を参照)。

Transport query(Transport);

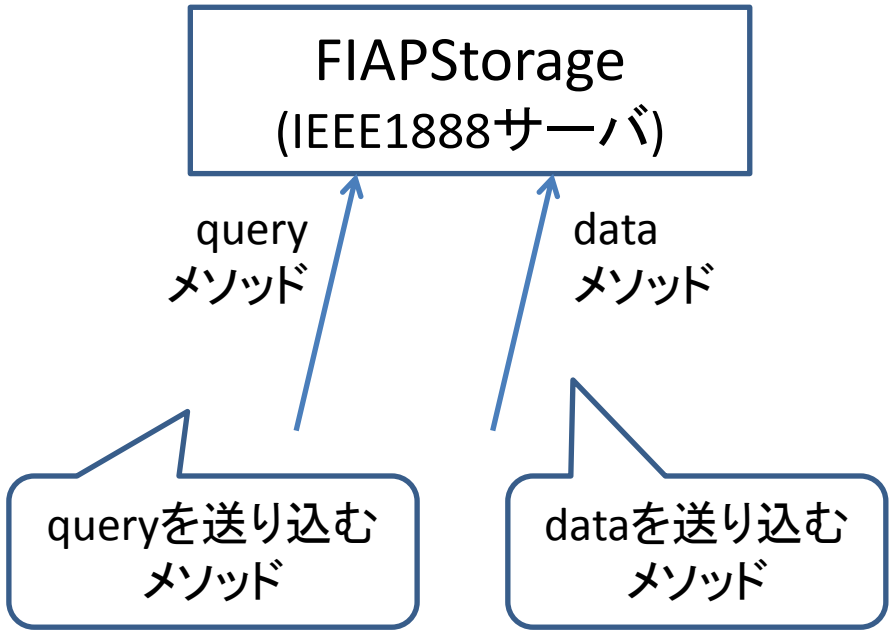
引数: 読出し要求

戻り値: 読出し結果応答

Transport data(Transport);

引数: 書込み要求

戻り値: 書込みの成功・失敗に関する情報



(注) FETCH, WRITE, TRAPはこれらを組み合わせた通信手順として規定されている

3.2. XML の基本構造

資料: IEEE1888 Storageとの通信仕様.pdf

IEEE1888 サーバとの通信で交わされる XML は、次のような構造を持つ。

---------- XML の基本構造(ここから) ----------

```
<transport>
  <header>
    ① クエリに関する記述
    ② 要求の成功
    ③ 要求の失敗(エラー情報)
  </header>
  <body>
    ④ データに関する記述
  </body>
</transport>
```

---------- XML の基本構造(ここまで) ----------

TCP上の通信例(キャプチャ)は、
同資料の付録Bを参照のこと！

(注) 実際には、XMLを意識せず
プログラミング可能です。
(重要なのはオブジェクトの構造)

この XML は、query メソッドおよび data メソッドの呼び出し時と応答時にやり取りされる。ここで header 部には「①クエリに関する記述」、「②要求の成功」、「③要求の失敗(エラー情報)」が記載され、body 部には「④データに関する記述」が記載される。

FETCH 最新値取得(リクエスト)の XML文書構造

資料:「IEEE1888プログラミング・スタートアップ・マニュアル.pdf」-- 第3.1章

①クエリに関する記述

```
-- FETCH最新値リクエスト例(ここから)--
<transport xmlns="http://gutp.jp/fiap/2009/11/">
 <header>
  <query id="0a90f1fa-bdb4-48ff-87d3-661d2af6ff4c" type="storage">
   <key id="http://www.gutp.jp/dummy/integer" attrName="time" select="maximum"/>
   <key id="http://www.gutp.jp/dummy/real1" attrName="time" select="maximum"/>
  </query>
 </header>
</transport>
-- FETCH最新値リクエスト例(ここまで)--
```

このURIで示される(センサ)の　最新値を要求

(*) ポイントIDと呼ぶ・・・詳細は後述

⇨ クエリの記述方法は、「IEEE1888 Storageとの通信仕様.pdf」の第3.3章および第5章を参照のこと

FETCH 最新値取得(レスポンス)のXML文書構造

資料:「IEEE1888プログラミング・スタートアップ・マニュアル.pdf」-- 第3.1章

```
-- FETCH最新値レスポンス例(ここから)--
<transport xmlns="http://gutp.jp/fiap/2009/11/">
 <header>
  <OK/>
  <query id="0a90f1fa-bdb4-48ff-87d3-661d2af6ff4c" type="storage">
   <key id="http://www.gutp.jp/dummy/integer" attrName="time" select="maximum"/>
   <key id="http://www.gutp.jp/dummy/real1" attrName="time" select="maximum"/>
  </query>
 </header>
 <body>
  <point id="http://www.gutp.jp/dummy/integer">
   <value time="2011-09-03T11:35:00.000+09:00">76419323</value>
  </point>
  <point id="http://www.gutp.jp/dummy/real1">
   <value time="2011-09-03T11:35:00.000+09:00">-99.8</value>
  </point>
 </body>
</transport>
-- FETCH最新値レスポンス例(ここまで)--
```

②要求の成功・③失敗 に関する記述

④データに関する記述

(FETCHの場合は読み出されたデータ)

データの記述方法は、「IEEE1888 Storageとの通信仕様.pdf」の第4章を参照のこと

プログラミングの準備

- SDKの立ち上げ
  - SDKスタートアップマニュアルの 2.1. を参照
- SDKのIPアドレスを確認
  - SDKスタートアップマニュアルの 2.5. を参照
- SOAP通信スタブの作成
  - Visual Studioの場合
    - プログラミングマニュアルの 3.2.1. を参照
  - Javaの場合
    - プログラミングマニュアルの 3.3.1. および 3.3.2. を参照
  - PHP5の場合
    - プログラミングマニュアルの 3.4.1. を参照

簡単に始めたい人にはこちらがお勧め

本日(15日)のスケジュール

10:00　IEEE1888とは何か？（全体像）

11:00　IEEE1888機器を使ってみる

12:00　昼食

13:00　IEEE1888ソフトウェア開発キット(SDK)の概要

13:30　SDKを使った FETCH手順のプログラミング

15:00　休憩

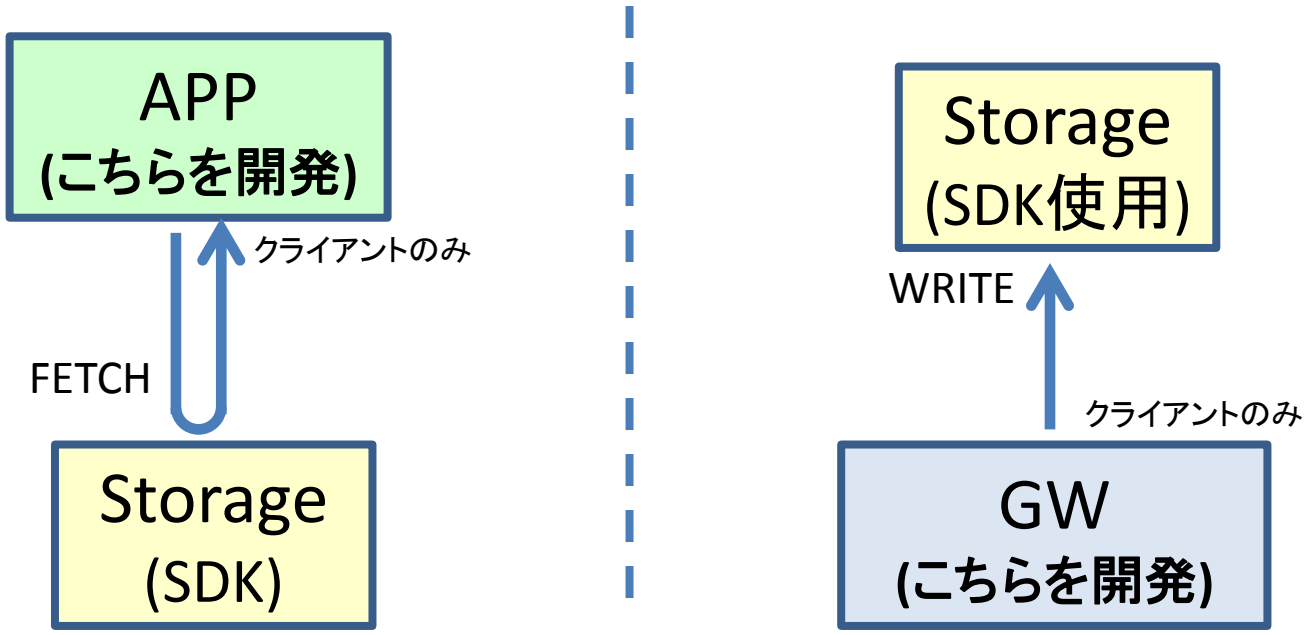
15:30　SDKを使った WRITE手順のプログラミング

16:30　C言語版/Arduino版についての紹介

17:00　解散

本実践ワークショップでの演習パターン

- FETCHプログラミング演習
- WRITEプログラミング演習



1．FETCHの演習

2．WRITEの演習

FETCHのプログラミング演習

- 作成するプログラム
  - FIAPStorageサーバからデータを読み出して表示するプログラム（最新値＆過去データの読出し）
- 演習の狙い
  - FETCH手順をプログラミングレベルから理解する
- この技術を習得すると
  - 「見える化」アプリケーションを作成することができるようになる (もちろん、これに限らない)

東京大学
工学部2号館
江崎研究室
ステータス

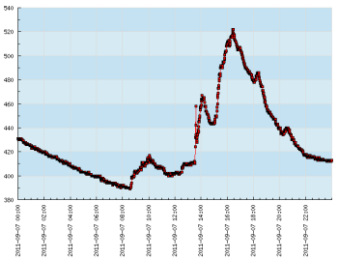
環境(学生室)
環境(教授室)
空調設備
電力系統

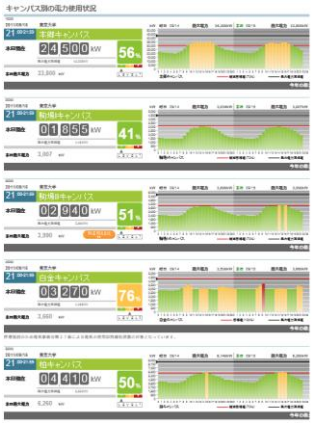
江崎研 学生室の環境

取得時刻: 2011-09-08 09:03:41

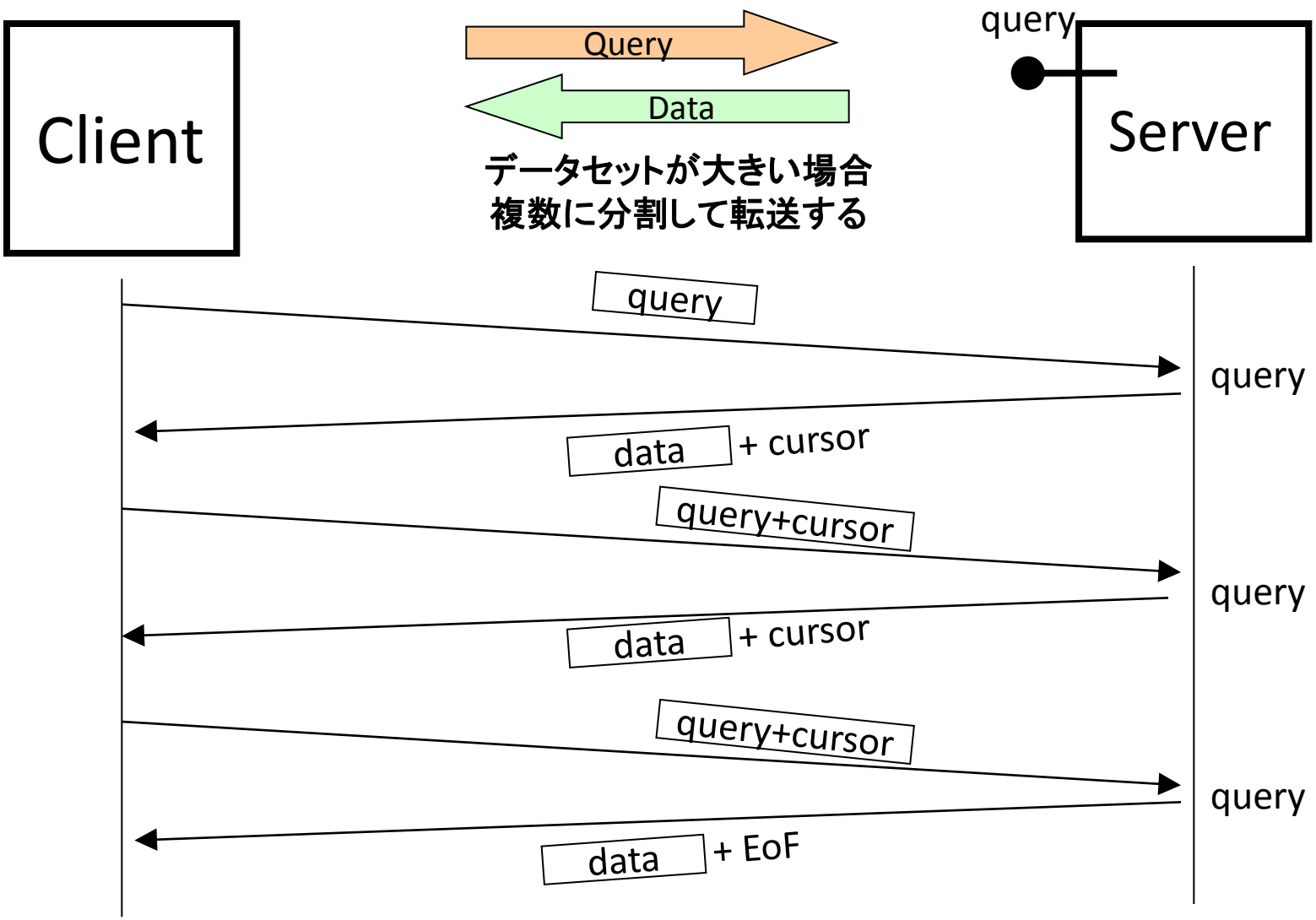
場所詳細	センサタイプ	温度(℃)	湿度(%)	CO2濃度(ppm)	照度(Lux)	気圧(hPa)
入り口付近	HCUCO2	27.89	56.75	443.00	167.41	1007.59
Kino付近	山武無線センサ	25	63	--	--	--
Lucho付近	山武無線センサ	25.2	64	--	--	--
Thomas付近	山武無線センサ	26.5	62	--	--	--
Shou付近	山武無線センサ	25	64	--	--	--

(*) 過去180秒以内に更新されていない項目は　で表示されます。





FETCH手順の解説


Client
Query
Data
データセットが大きい場合
複数に分割して転送する
query
Server
query
query
data
+ cursor
query+cursor
query
data
+ cursor
query+cursor
query
data
+ EoF

FETCH手順 (最初のリクエスト)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<transport xmlns="http://gutp.jp/fiap/2009/11/">
<header>
<query acceptableSize="10" id="1024f06b-b3f5-4863-a497-e6b732f26d75" type="storage">
<key attrName="time" gteq="2010-01-06T18:00:00+09:00" id="http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/02F/221-2/DB" lt="2010-01-
07T00:00:00.0000000+09:00"/>
</query>
</header>
</transport>
```

クエリ

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<transport xmlns="http://gutp.jp/fiap/2009/11/">
<header>
<OK/>
<query acceptableSize="1" cursor="494d6558-0005-4b1c-b22d-9897af70418b" id="1024f06b-b3f5-4863-a497-e6b732f26d75" type="storage">
<key attrName="time" gteq="2010-01-06T18:00:00.0000000+09:00" id="http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/02F/221-2/DB" lt="2010-01-
07T00:00:00+09:00"/>
</query>
</header>
<point id="http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/02F/221-2/DB">
<value time="2010-01-06T18:00:00+09:00">25.0</value>
<value time="2010-01-06T18:01:00+09:00">25.0</value>
<value time="2010-01-06T18:02:00+09:00">25.0</value>
<value time="2010-01-06T18:03:00+09:00">25.0</value>
<value time="2010-01-06T18:04:00+09:00">25.0</value>
<value time="2010-01-06T18:05:00 +09:00">24.75</value>
<value time="2010-01-06T18:06:00+09:00">24.75</value>
<value time="2010-01-06T18:07:00+09:00">24.75</value>
<value time="2010-01-06T18:08:00+09:00">24.75</value>
<value time="2010-01-06T18:09:00+09:00">24.75</value>
</point>
</transport>
```

カーソル

読み出されたデータ (subset)

FETCH手順 (2回目のリクエスト)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<transport xmlns="http://gutp.jp/fiap/2009/11/">
<header>
<query acceptableSize="1[illegible]" cursor="494d6558-0005-4b1c-b22d-9897af70418b" id="1024f06b-b3f5-4863-a497-e6b732f26d75" type="storage">
<key attrName="time" gteq="[illegible]" id="http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/02F/221-2/DB" lt="2010-01-
07T00:00:00+09:00"/>
</query>
</header>
</transport>
```

カーソル

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<transport xmlns="http://gutp.jp/fiap/2009/11/">
<header>
<OK/>
<query acceptableSize="1[illegible]" cursor="494d6558-0005-4b1c-b22d-9897af70418b" id="1024f06b-b3f5-4863-a497-e6b732f26d75" type="storage">
<key attrName="time" gteq="[illegible]" id="http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/02F/221-2/DB" lt="2010-01-
07T00:00:00+09:00"/>
</query>
</header>
<body>
<point id="http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/02F/221-2/DB">
<value time="2010-01-06T18:10:00+09:00">24.75</value>
<value time="2010-01-06T18:11:00+09:00">24.75</value>
<value time="2010-01-06T18:12:00+09:00">24.75</value>
<value time="2010-01-06T18:13:00+09:00">24.75</value>
<value time="2010-01-06T18:14:00+09:00">24.75</value>
<value time="2010-01-06T18:15:00+09:00">24.75</value>
<value time="2010-01-06T18:16:00+09:00">24.75</value>
<value time="2010-01-06T18:17:00+09:00">24.75</value>
<value time="2010-01-06T18:18:00+09:00">24.75</value>
<value time="2010-01-06T18:19:00+09:00">24.75</value>
</point>
</body>
</transport>
```

カーソル

FETCH手順 (最終回のリクエスト)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<transport xmlns="http://gutp.jp/fiap/2009/11/">
<header>
<query acceptableSize="10" cursor=" 494d6558-0005-4b1c-b22d-9897af70418b " id="1024f06b-b3f5-4863-a497-e6b732f26d75" type="storage">
<key attrName="time" gteq="2010-01-06T18:00:00.0000000+09:00" id="http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/02F/221-2/DB" lt="2010-01-
07T00:00:00+09:00"/>
</query>
</header>
</transport>
```

カーソル

Response

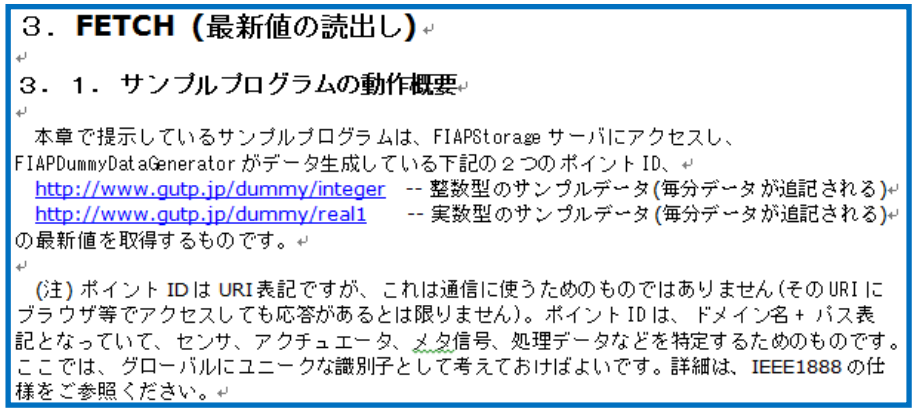
```
<?xml version="1.0" encoding="UTF-8"?>
<transport xmlns="http://gutp.jp/fiap/2009/11/">
<header>
<OK/>
<query acceptableSize="10" id="1024f06b-b3f5-4863-a497-e6b732f26d75" type="storage">
<key attrName="time" gteq="2010-01-06T18:00:00.0000000+09:00" id="http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/02F/221-2/DB" lt="2010-01-
07T00:00:00+09:00"/>
</query>
</header>
<body>
<point id="http://fiap-gw.gutp.ic.i.u-tokyo.ac.jp/EngBldg2/02F/221-2/DB">
<value time="2010-01-06T23:50:00+09:00">21.75</value>
<value time="2010-01-06T23:51:00+09:00">21.75</value>
<value time="2010-01-06T23:52:00+09:00">21.75</value>
<value time="2010-01-06T23:53:00+09:00">21.75</value>
<value time="2010-01-06T23:54:00+09:00">21.75</value>
<value time="2010-01-06T23:55:00+09:00">21.75</value>
<value time="2010-01-06T23:56:00+09:00">21.75</value>
<value time="2010-01-06T23:57:00+09:00">21.75</value>
<value time="2010-01-06T23:58:00+09:00">21.75</value>
<value time="2010-01-06T23:59:00+09:00">21.75</value>
</point>
</body>
</transport>
```

カーソルがない (= 終了の合図)

FETCH に関するプログラミング・マニュアル

資料: IEEE1888プログラミング・スタートアップ・マニュアル.pdf

第3章　最新値読出しのサンプル



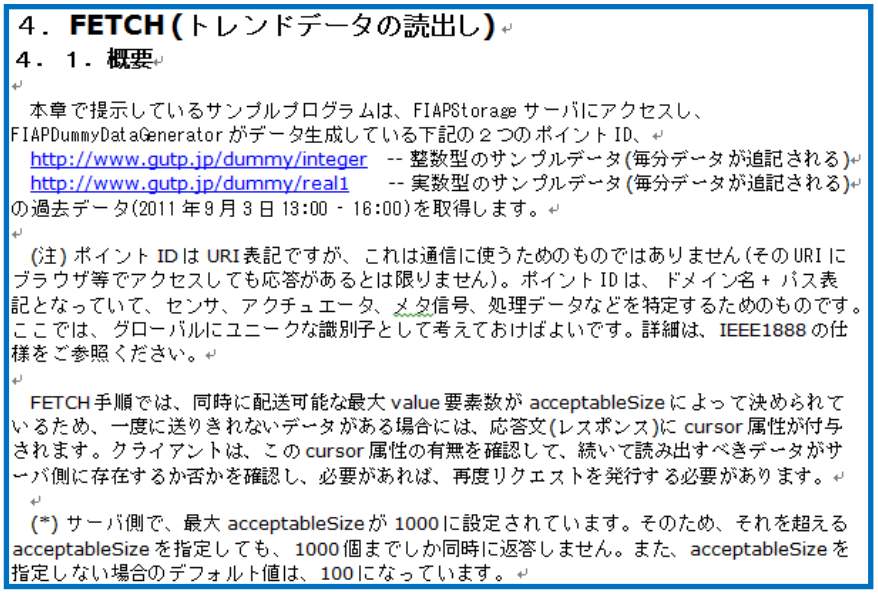
3．FETCH（最新値の読出し）

3．1．サンプルプログラムの動作概要

本章で提示しているサンプルプログラムは、FIAPStorage サーバにアクセスし、FIAPDummyDataGenerator がデータ生成している下記の２つのポイント ID、
http://www.gutp.jp/dummy/integer　-- 整数型のサンプルデータ(毎分データが追記される)
http://www.gutp.jp/dummy/real1　-- 実数型のサンプルデータ(毎分データが追記される)
の最新値を取得するものです。

(注) ポイント ID は URI 表記ですが、これは通信に使うためのものではありません(その URI にブラウザ等でアクセスしても応答があるとは限りません)。ポイント ID は、ドメイン名 + パス表記となっていて、センサ、アクチュエータ、メタ信号、処理データなどを特定するためのものです。ここでは、グローバルにユニークな識別子として考えておけばよいです。詳細は、IEEE1888 の仕様をご参照ください。

第4章　過去データ読出しのサンプル



4．FETCH(トレンドデータの読出し)

4．1．概要

本章で提示しているサンプルプログラムは、FIAPStorage サーバにアクセスし、FIAPDummyDataGenerator がデータ生成している下記の２つのポイント ID、
http://www.gutp.jp/dummy/integer　-- 整数型のサンプルデータ(毎分データが追記される)
http://www.gutp.jp/dummy/real1　-- 実数型のサンプルデータ(毎分データが追記される)
の過去データ(2011 年 9 月 3 日 13:00 - 16:00)を取得します。

(注) ポイント ID は URI 表記ですが、これは通信に使うためのものではありません(その URI にブラウザ等でアクセスしても応答があるとは限りません)。ポイント ID は、ドメイン名 + パス表記となっていて、センサ、アクチュエータ、メタ信号、処理データなどを特定するためのものです。ここでは、グローバルにユニークな識別子として考えておけばよいです。詳細は、IEEE1888 の仕様をご参照ください。

FETCH 手順では、同時に配送可能な最大 value 要素数が acceptableSize によって決められているため、一度に送りきれないデータがある場合には、応答文(レスポンス)に cursor 属性が付与されます。クライアントは、この cursor 属性の有無を確認して、続いて読み出すべきデータがサーバ側に存在するか否かを確認し、必要があれば、再度リクエストを発行する必要があります。

(*) サーバ側で、最大 acceptableSize が 1000 に設定されています。そのため、それを超える acceptableSize を指定しても、1000 個までしか同時に返答しません。また、acceptableSize を指定しない場合のデフォルト値は、100 になっています。

⇨ このマニュアルに従って演習開始

Visual C# の場合: 比較的簡単
Javaの場合: 事前準備が少々大変
PHP言語の場合: サンプルプログラムがSDKのデスクトップにある (楽)

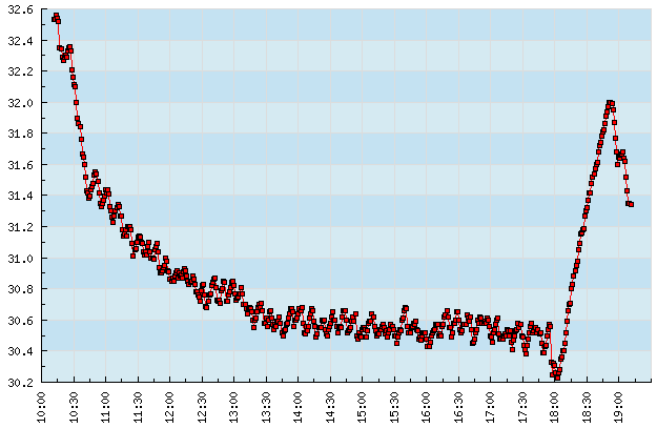
[重要] 実際のAPP開発においては設定を後から与えられるようにする

環境情報

取得時刻: 2011-09-17 13:13:28

設置場所	温度(℃)	照度(Lux)	DIPSW設定値	トグルSW状態
自宅	40.0	0	8	OFF
部屋A	27.8	334	15	OFF
部屋B	32.2	613	0	OFF
部屋C	29.7	334	11	OFF
部屋D	29.7	279	15	OFF

(*) 過去180秒以内に更新されていない項目は　で表示されます。



表形式への整理

グラフ化

IEEE1888サーバ

アプリケーション

インターネット

IEEE1888による通信

ソフトウェアデーモン
として実装されることが多い

動作設定

① サーバ設定
② データ設定
…. など

本日(15日)のスケジュール

10:00　IEEE1888とは何か？（全体像）

11:00　IEEE1888機器を使ってみる

12:00　昼食

13:00　IEEE1888ソフトウェア開発キット(SDK)の概要

13:30　SDKを使ったFETCH手順のプログラミング

15:00　休憩

15:30　SDKを使ったWRITE手順のプログラミング

16:30　C言語版/Arduino版についての紹介

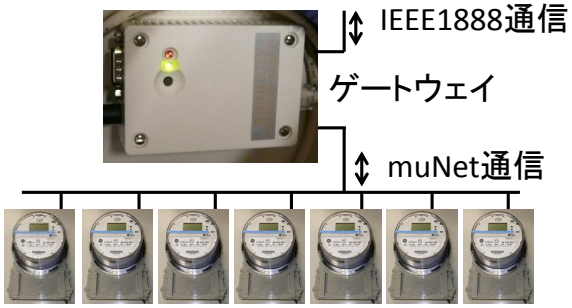
17:00　解散

WRITEのプログラミング演習

- 作成するプログラム
  - FIAPStorageサーバにデータを送信(格納)するプログラム
- 演習の狙い
  - WRITE手順をプログラミングレベルから理解する
- この技術を習得すると
  - 各種変換GWを作成することができるようになる
  - 各種センサからの情報収集ができるようになる
    (これらに限らない)

Lonworks GW

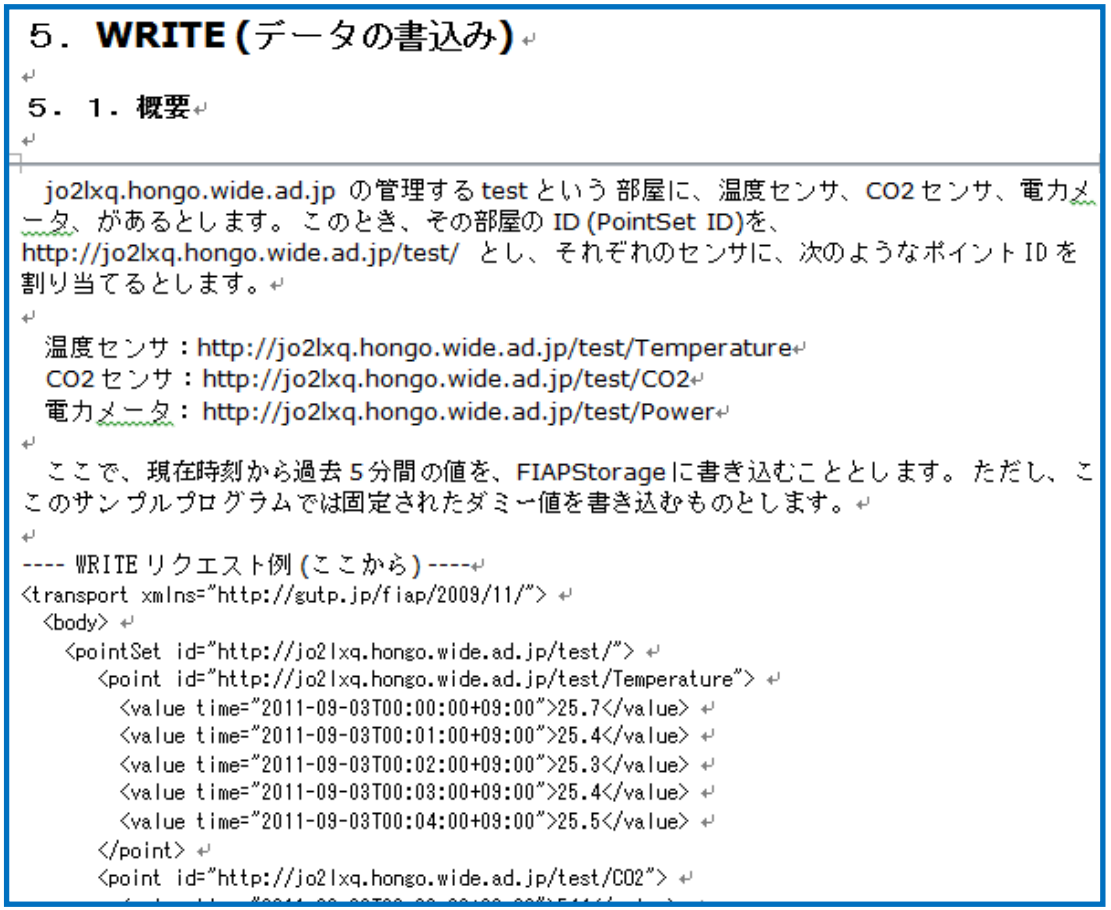

BACnet/IP GW
(Armadillo-210)



GE社の電力メータGW

WRITEに関する
プログラミング・マニュアル

資料: IEEE1888プログラミング・スタートアップ・マニュアル.pdf



５．WRITE (データの書込み)

５．１．概要

jo2lxq.hongo.wide.ad.jp の管理する test という 部屋に、温度センサ、CO2 センサ、電力メータ、があるとします。 このとき、その部屋の ID (PointSet ID)を、
http://jo2lxq.hongo.wide.ad.jp/test/ とし、それぞれのセンサに、次のようなポイント ID を割り当てるとします。

温度センサ：http://jo2lxq.hongo.wide.ad.jp/test/Temperature
CO2 センサ：http://jo2lxq.hongo.wide.ad.jp/test/CO2
電力メータ： http://jo2lxq.hongo.wide.ad.jp/test/Power

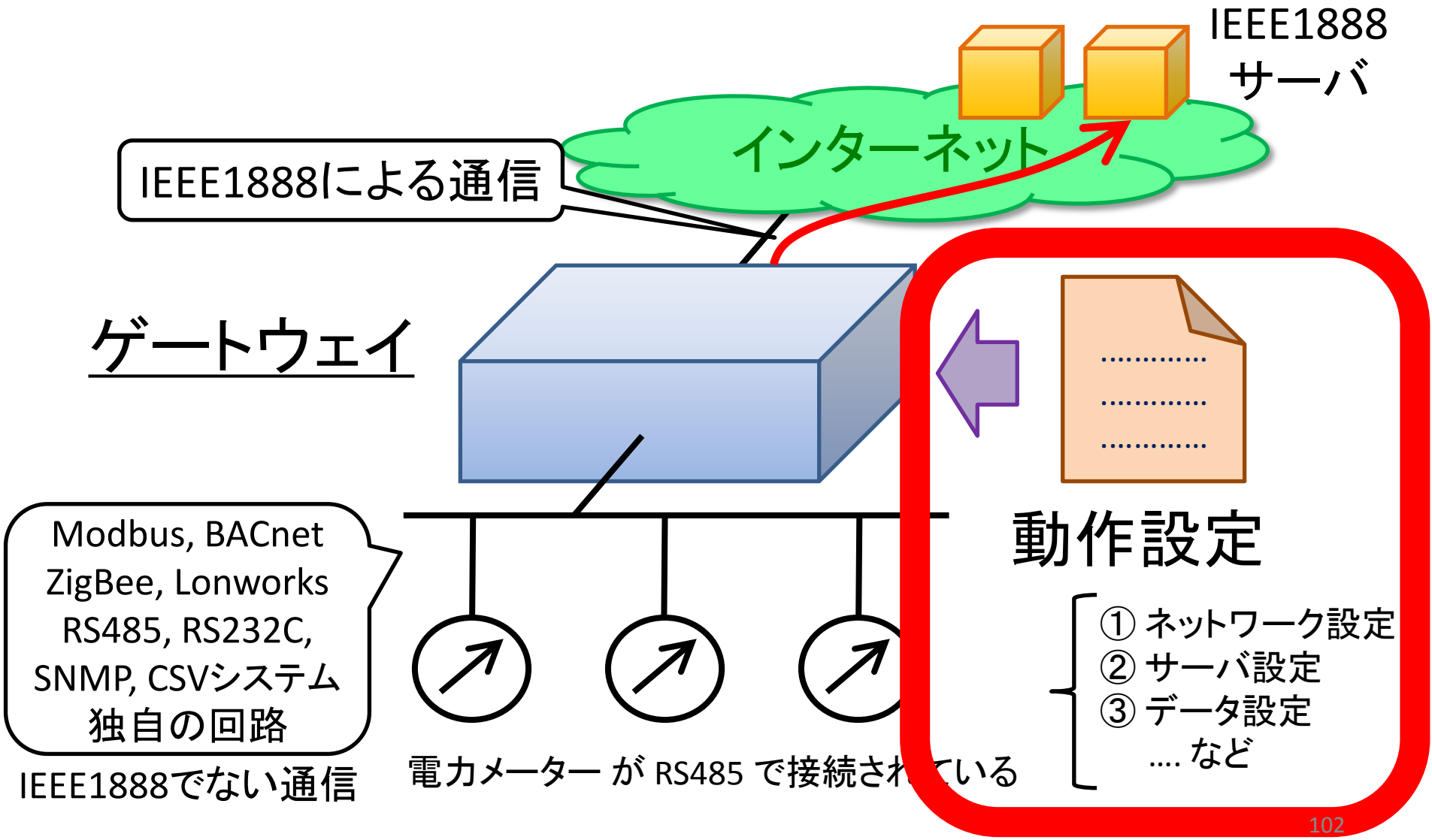
ここで、現在時刻から過去 5 分間の値を、FIAPStorage に書き込むこととします。ただし、このサンプルプログラムでは固定されたダミー値を書き込むものとします。

---- WRITE リクエスト例 (ここから) ----

```
<transport xmlns="http://gutp.jp/fiap/2009/11/">
  <body>
    <pointSet id="http://jo2lxq.hongo.wide.ad.jp/test/">
      <point id="http://jo2lxq.hongo.wide.ad.jp/test/Temperature">
        <value time="2011-09-03T00:00:00+09:00">25.7</value>
        <value time="2011-09-03T00:01:00+09:00">25.4</value>
        <value time="2011-09-03T00:02:00+09:00">25.3</value>
        <value time="2011-09-03T00:03:00+09:00">25.4</value>
        <value time="2011-09-03T00:04:00+09:00">25.5</value>
      </point>
      <point id="http://jo2lxq.hongo.wide.ad.jp/test/CO2">
```

⇨ このマニュアルに従って演習続行

[重要] 実際のGW開発においては設定を後から与えられるようにする



本日(15日)のスケジュール

10:00　IEEE1888とは何か？（全体像）

11:00　IEEE1888機器を使ってみる

12:00　昼食

13:00　IEEE1888ソフトウェア開発キット(SDK)の概要

13:30　SDKを使ったFETCH手順のプログラミング

15:00　休憩

15:30　SDKを使ったWRITE手順のプログラミング

16:30　C言語版/Arduino版についての紹介

17:00　解散

IEEE1888通信スタック (C言語版)

- GW機器は、主に 組込みPC に実装される
- 現実的な価格の組込みPCでは、C言語での開発が主流

Armadillo-210 (Atmark Techno)

OpenWrt

などなど

- IEEE1888通信スタック (双方向版)
  - 組込みPCでのGW開発を支援するために作られた
  - オープンソース (BSDライセンス下で配布)

IEEE1888通信スタック (C言語版)

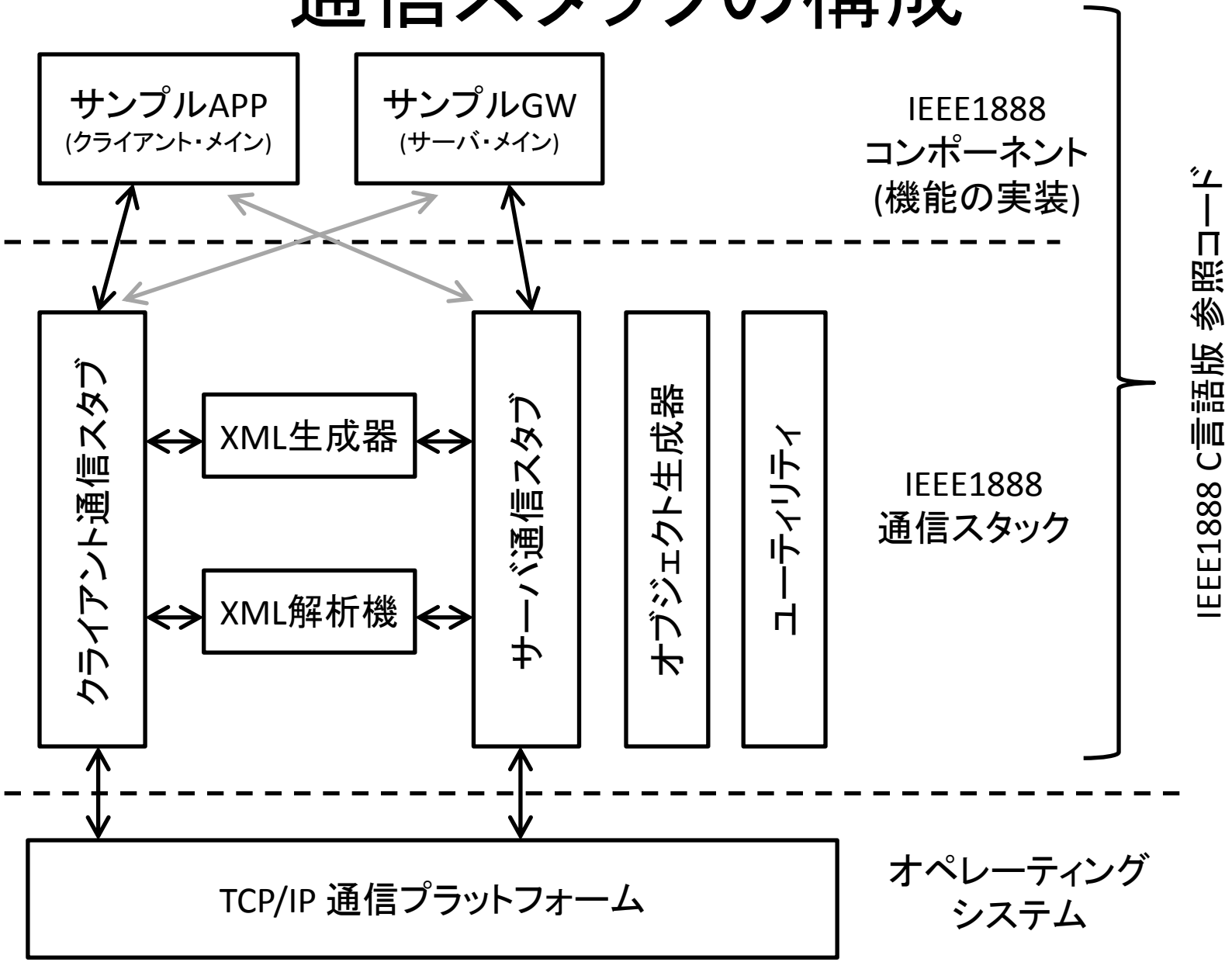
- 入手方法
  - http://gutp.jp/fiap/kit.html (参照コード: C言語版) よりダウンロード

ソフトウェアの構成

ファイル名	内容
ieee1888.h	メインのヘッダ・ファイル
ieee1888_client.c	IEEE1888のquery/dataメソッドのクライアント通信スタブ
ieee1888_object_factory.c	IEEE1888オブジェクト(transportやbody等)の生成工場
ieee1888_sample_gw.c	GWのサンプル実装 (サーバ実装例)
ieee1888_sample_app.c	APPのサンプル実装 (クライアント実装例)
ieee1888_server.c	IEEE1888のquery/dataメソッドのサーバ通信スタブ
ieee1888_util.c	便利な機能(オブジェクトのダンプ表示等)の集合
ieee1888_XMLgenerator.h	XMLシリアル化のためのヘッダ・ファイル
ieee1888_XMLgenerator.c	IEEE1888オブジェクトのXMLシリアル化機能(生成器)実装
ieee1888_XMLparser.h	XMLデシリアル化のためのヘッダ・ファイル
ieee1888_XMLparser.c	IEEE1888オブジェクトのXMLデシリアル化機能(解析機)実装

IEEE1888通信スタック (C言語版)

通信スタックの構成



IEEE1888通信スタック (C言語版)
プログラム例: オブジェクトの生成

```
/* IEEE1888要求メッセージ(オブジェクト)の生成・整理 */
  ieee1888_transport* request=ieee1888_mk_transport();
  request->body=ieee1888_mk_body();

  ieee1888_pointSet* ps=ieee1888_mk_pointSet_array(1);
  ps->id=ieee1888_mk_uri("http://www.gutp.jp/dummy/");
  request->body->pointSet=ps;
  request->body->n_pointSet=1;

  ieee1888_point* p=ieee1888_mk_point_array(3);
  ps->point=p;
  ps->n_point=3;

  /* 送信するデータをIEEE1888要求メッセージ(オブジェクト)に埋め込む */
  for(i=0;i<3;i++){
    p[i].id=ieee1888_mk_uri(mydata[i].id);
    p[i].value=ieee1888_mk_value();
    p[i].n_value=1;
    p[i].value->time=ieee1888_mk_time(mydata[i].time);
    p[i].value->content=ieee1888_mk_string(mydata[i].value);
  }
```

```
/* 送信するセンサーデータの格納用 */
struct app_data {
char id[100];
time_t time;
char value[100];
};

 struct app_data mydata[3];

 /* ポイントID設定 */
 strcpy(mydata[0].id,"http://www.gutp.jp/dummy/point0");
 strcpy(mydata[1].id,"http://www.gutp.jp/dummy/point1");
 strcpy(mydata[2].id,"http://www.gutp.jp/dummy/point2");
```

IEEE1888通信スタック (C言語版)
プログラム例: クライアント＆サーバ

※ dataメソッドを呼出すクライアントの例

```
ieee1888_transport* response=ieee1888_client_data(request,
    "http://fiap-sandbox.gutp.ic.i.u-tokyo.ac.jp/axis2/services/FIAPStorage",NULL,&err);
```

※ dataメソッドを呼出されるサーバの例

```
/* dataメソッドのサービスハンドラ実装 */
ieee1888_transport* ieee1888_server_data(ieee1888_transport* request,char** args){
  …
  /* OK応答を返す */
  ieee1888_transport* response=ieee1888_mk_transport();
  response->header=ieee1888_mk_header();
  response->header->OK=ieee1888_mk_OK();
  return response;
}

 /* dataメソッドのハンドラ(今回は上記のieee1888_server_data)を登録する */
 ieee1888_set_service_handlers(NULL,ieee1888_server_data);

 /* サーバを開始する */
 ieee1888_server_create(1888);
```

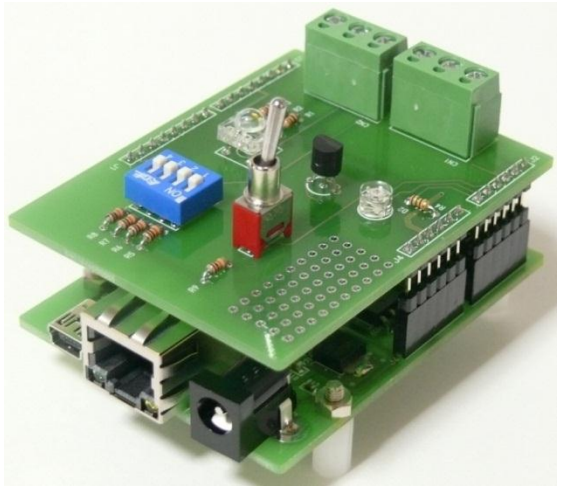
IEEE1888通信スタック (Arduino版)

- Arduino とは (http://www.arduino.cc より引用)
  - 原文: Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.
  - 和訳: Arduinoは、簡単に使える柔軟なハードウェアとソフトウェアのオープンソースな電子工作プラットフォームです。芸術家、デザイナー、愛好家ほか、インタラクティブな機器や環境を作り出すことに興味がある人向けに開発されたものです。

http://www.arduino.cc より引用

- 超高速なアジャイル開発が得意
- 様々な事業で使われている
  オープンなプラットフォーム
- 2011年8月、このボード上に簡易的なIEEE1888通信スタックを実装できることが証明された

IEEE1888通信スタック (Arduino版) の応用例

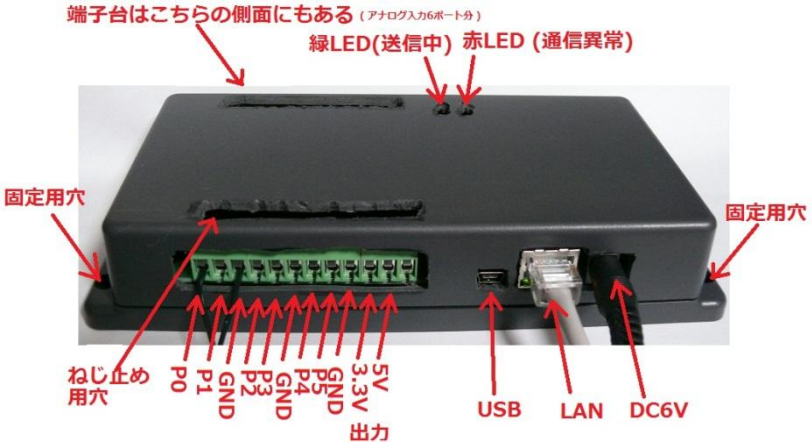


IEEE1888学習キット

ピークカットヘルパー

その他:

1. ZigBee (スマートメータ) からのデータ収集
2. ZigBee (人感センサ) からのデータ収集
3. CTによるタップ電力計測 など。。



開発進行中の機器

・パルス計数

・汎用デジタル・アナログ入力

・汎用デジタル出力

など。。。

IEEE1888通信スタック (Arduino版)

- 入手先
  - http://gutp.jp/fiap/kit.html

- FIAPUploadAgent
  - WRITEクライアント機能を実装

- FIAPDownloadAgent
  - FETCHクライアント機能を実装

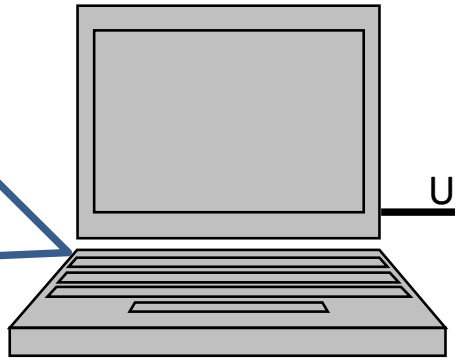
IEEE1888通信スタック (Arduino版)
ソフトウェアのプログラミング方法

ソフトウェアをパソコン(Arduino IDE)で開発し、USBでインストールする。

Arduino統合開発環境(IDE)
ターゲットボード名: Arduino UNO

Arduino IDEを使ってプログラムを作成＆
コンパイルし、IEEE1888開発ボードに
インストールします。

200行～1000行のC, C++ のプログラムで
IEEE1888のGWノードを実装できます。



開発用PC

IEEE1888
ネットワーク

Ethernet

USB

IEEE1888開発ボード

各種センサ
(例: 照度センサ)

各種通信媒体
(例: ZigBee)

プログラミング後は、パソコンは不要(スタンドアロンで動作する)。

サンプルコードは http://gutp.jp/fiap/kit.html からダウンロードできる。

明日(16日)のスケジュール

10:00　IEEE1888レジストリの仕組み

10:30　レジストリを用いたCEMS実現への取組み

11:00　IEEE1888レジストリとの通信プログラミング実習

12:00　昼食

13:00　IEEE1888システムの設計と構築: その1

14:30　休憩

15:00　IEEE1888システムの設計と構築: その2

16:00　簡易な相互接続実験

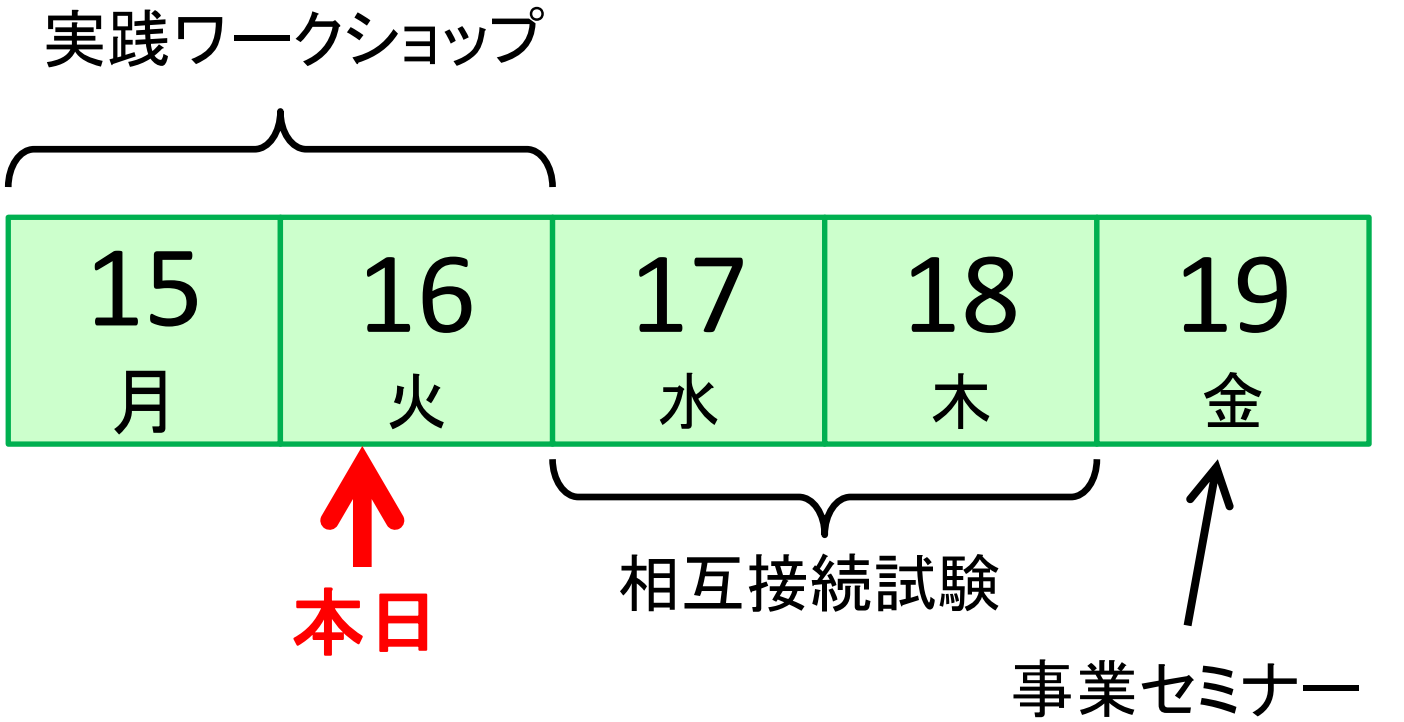
17:00　解散

IEEE1888実践ワークショップ
～２日目～

主催: 東大グリーンICTプロジェクト
　　プロトコル標準化WG
場所: 東京大学 工学部2号館 電気系会議室５
日程: 2012年10月15日-16日

はじめに

- 今週(15日-19日)は IEEE1888 Week です。



http://www.ssk21.co.jp/seminar/S_12347.html

はじめに

- 実践ワークショップ開催の目的
  - IEEE1888 に関連する機器開発、ソフトウェア開発、サービス開発のノウハウを、一堂に会して共有し、各社での採用検討・製品化への足掛かりとする。

- カリキュラム
  - 第1日目
    - IEEE1888の背景・技術・実現事例について理解する
    - SDK(ソフトウェア開発キット) を使って IEEE1888プログラミングを理解する
  - 第2日目
    - 大規模 M2M (Machine-to-Machine) システム の実現について理解する
    - IEEE1888システムの設計・構築について理解する

昨日(15日)のスケジュール

10:00　IEEE1888とは何か？（全体像）

11:00　IEEE1888機器を使ってみる

12:00　昼食

13:00　IEEE1888ソフトウェア開発キット(SDK)の概要

13:30　SDKを使ったFETCH手順のプログラミング

15:00　休憩

15:30　SDKを使ったWRITE手順のプログラミング

16:30　C言語版/Arduino版についての紹介

17:00　解散

本日(16日)のスケジュール

10:00　IEEE1888レジストリの仕組み

10:30　レジストリを用いたCEMS実現への取組み

11:00　IEEE1888レジストリとの通信プログラミング実習

12:00　昼食

13:00　IEEE1888システムの設計と構築：その１

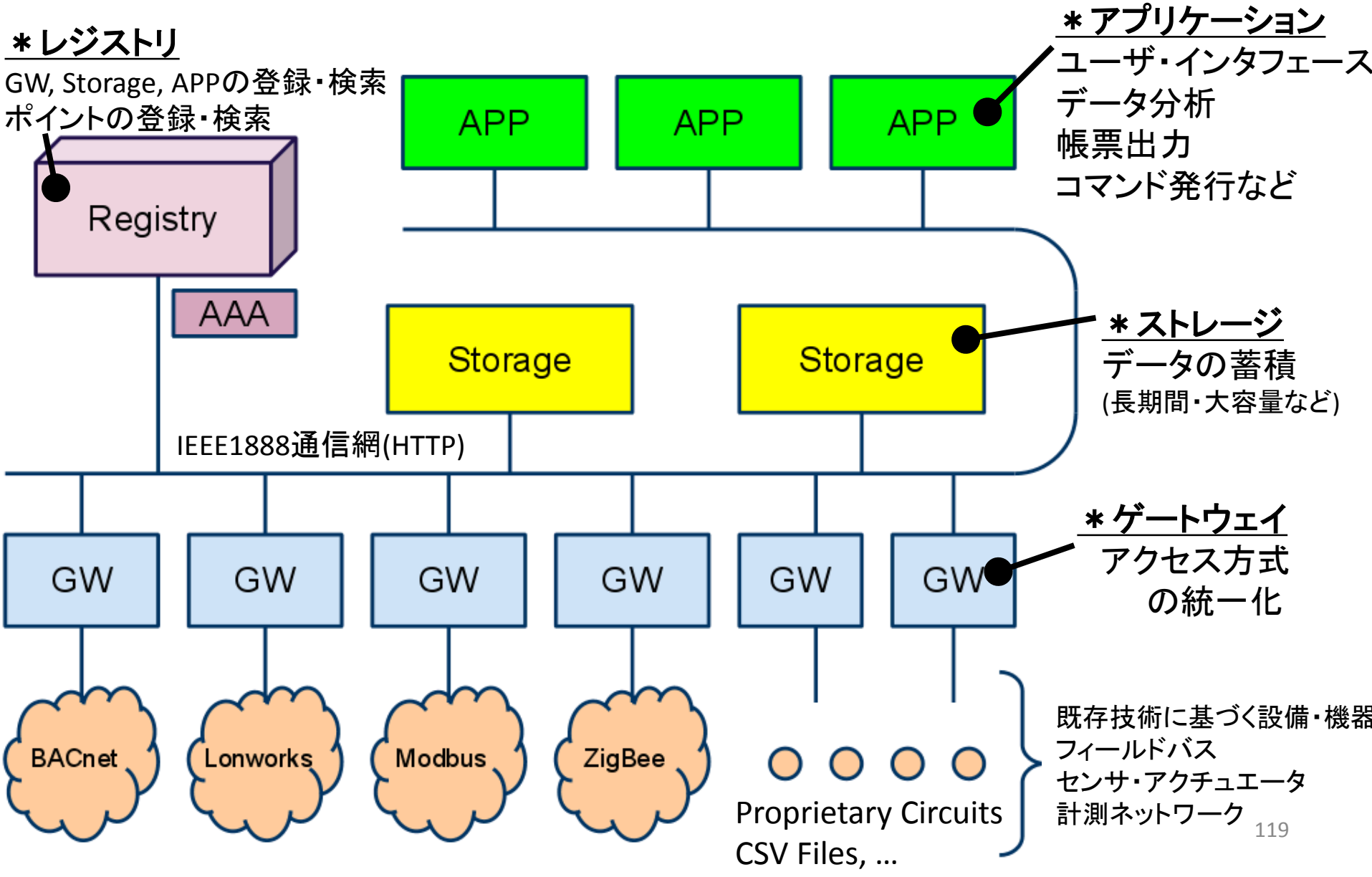
14:30　休憩

15:00　IEEE1888システムの設計と構築：その２

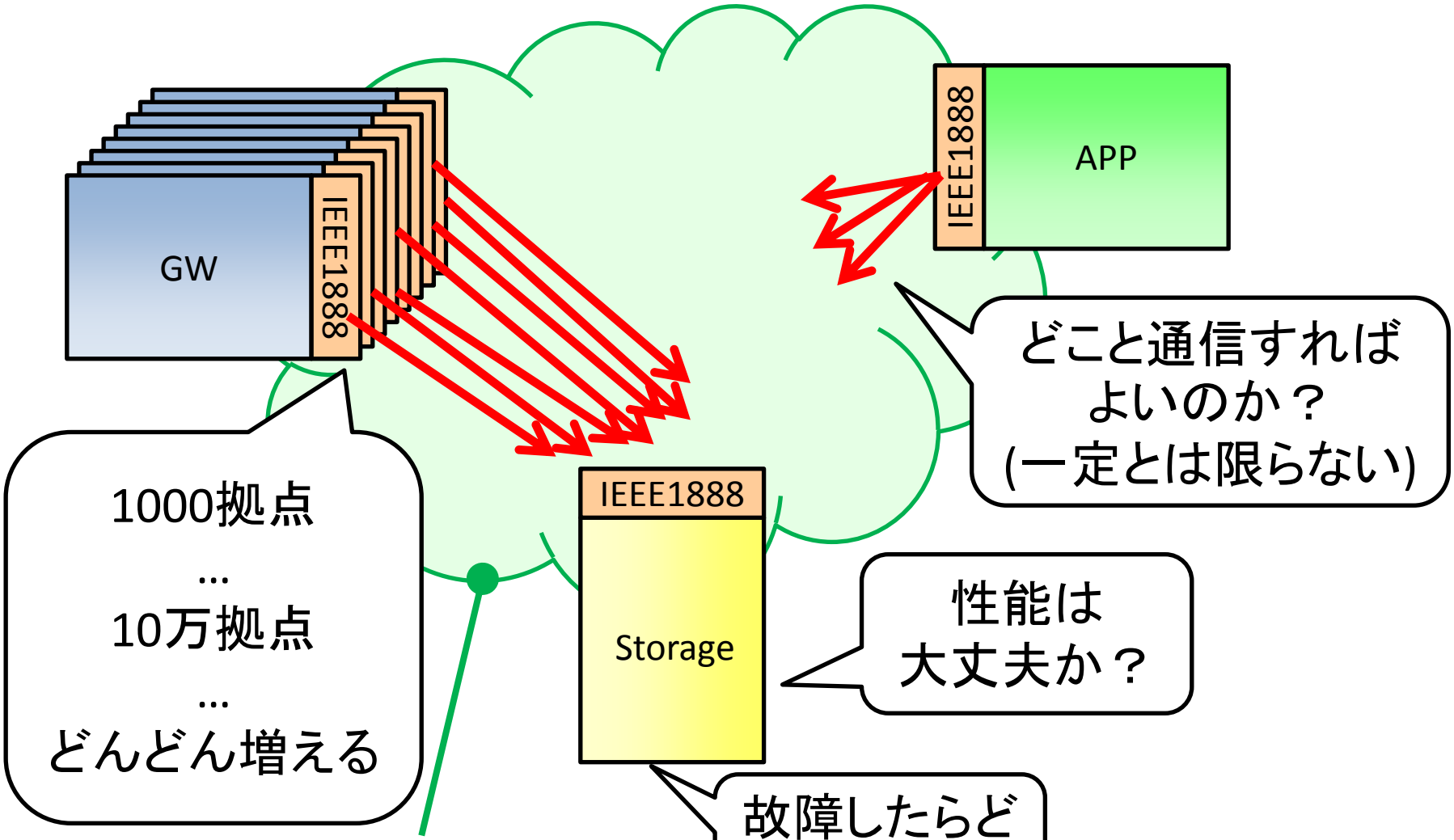
16:00　簡易な相互接続実験

17:00　解散

復習：IEEE1888 システム・アーキテクチャ



大規模化・高信頼運用を考える



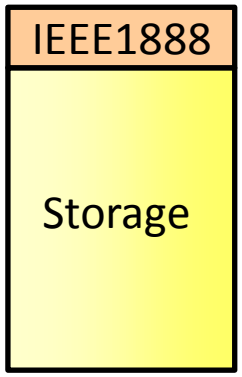
これらの課題を解決するための
ネットワーク的な仕組みが必要

システム大規模化への挑戦
2つのアプローチ

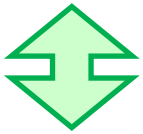
- スケールアップ方式
  - 高性能なコンピュータを用意する
    - 例1: 搭載メモリを 4GByte → 64GByteにする
  - 急激に高価になる

- スケールアウト方式
  - 複数のコンピュータに処理を分担させる
  - 数で勝負
  - 単価の安いコンピュータで済む

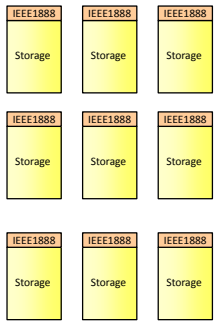
<u>※ IEEE1888はスケールアウト方式に対応!!</u>



巨大なサーバを
用意する



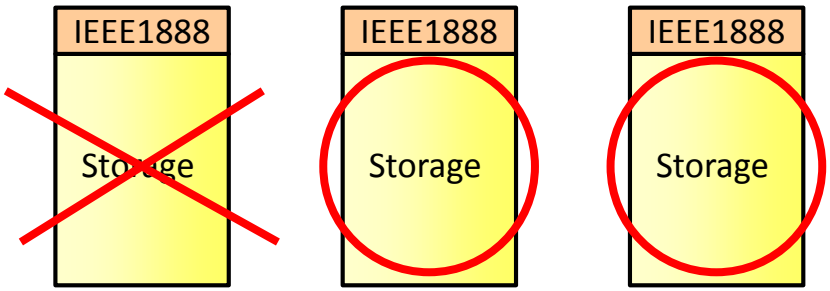
同等の性能
を有する



小さなサーバを
多数用意する

システム高信頼運用への挑戦

- 可用性の向上
  - 究極の目標: 24時間 365日 サービスを持続させたい
  - 実際: メンテナンスによる停止、故障による停止が発生する
- 冗長構成とその管理 (サービスを持続させるには)
  - 複数台でサービス提供
  - 1台が停止してもサービスは持続できる

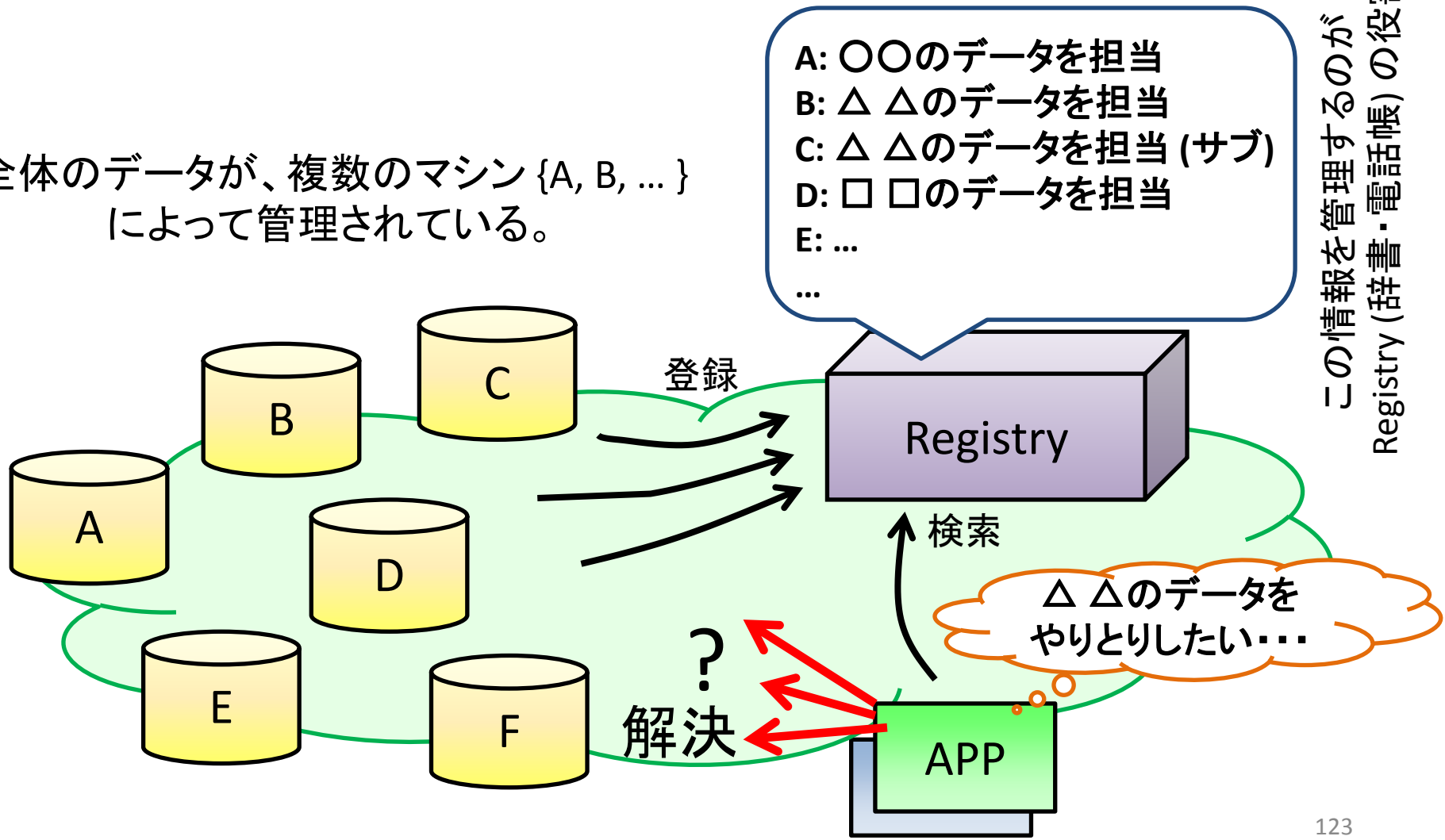


同一サービスを、複数台で提供する。
1台が故障しても、残りのマシンで、
サービスは提供し続けられる。

ただし、複数のマシンでサービス提供していることを伝える必要がある

レジストリの役割：その1
各コンポーネントの担当範囲を管理する

全体のデータが、複数のマシン {A, B, … }
によって管理されている。



コンポーネント対レジストリ通信の必要性

- コンポーネント対コンポーネント通信
  - (センサや制御)データ本体を交換する通信
    - GW ⇔ GW間、GW⇔ Storage間、GW⇔APP間
    - Storage⇔ Storage間、Storage⇔APP間
    - APP⇔APP間
  - WRITE, FETCH, TRAP手順が規定されている

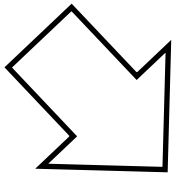
これだけでもIEEE1888システムの運用は出来る

- コンポーネント対レジストリ通信
  - システムの分散運用管理に関する通信
    - GW ⇔ Registry間、Storage⇔ Registry間、APP⇔ Registry間
  - REGISTRATION手順、LOOKUP手順が規定されている

(*) IEEE1888コンポーネントとは、GW, Storage, Appの総称
IEEE1888レジストリは、IEEE1888コンポーネントには属しない。

コンポーネントの担当範囲 ＆アクセス方法の記述

```
<component name="storage1" expires="600"
        priority="10" support="FETCH|WRITE"
        uri="http://133.11.168.3/axis2/services/FIAPStorage" >
 <key id="http://gutp.jp/pointA" … gteq="2012-01-01T00:00:00+09:00" />
 <key id="http://gutp.jp/pointB" … gteq="2012-01-01T00:00:00+09:00" />
 <key id="http://gutp.jp/pointC" … gteq="2012-01-01T00:00:00+09:00" />
 <key id="http://gutp.jp/pointD" … gteq="2012-01-01T00:00:00+09:00" />
 <key id="http://gutp.jp/pointE" … gteq="2012-01-01T00:00:00+09:00" />
</component>
```



この情報を Registry に登録する

REGISTRATION手順

- Registryの持つ registrationメソッドを使い、コンポーネントの担当情報を登録する

※ リクエストメッセージの例

```
<transport xmlns="http://gutp.jp/fiap-mgmt/2009/11/">
 <body>
  <component name="storage1" expires="600"
          priority="10" support="FETCH|WRITE"
          uri="http://133.11.168.3/axis2/services/FIAPStorage" >
   <key id="http://gutp.jp/pointA" … gteq="2012-01-01T00:00:00+09:00" />
   <key id="http://gutp.jp/pointB" … gteq="2012-01-01T00:00:00+09:00" />
   <key id="http://gutp.jp/pointC" … gteq="2012-01-01T00:00:00+09:00" />
   <key id="http://gutp.jp/pointD" … gteq="2012-01-01T00:00:00+09:00" />
   <key id="http://gutp.jp/pointE" … gteq="2012-01-01T00:00:00+09:00" />
  </component>
 </body>
</transport>
```

この情報は、あくまで600秒間有効なだけなので、
定期的にこの登録操作を実施する必要がある。

LOOKUP手順 (リクエスト)

- Registryが管理しているコンポーネント(対象となるデータを担当しているサーバ)の情報を「lookupメソッドを呼ぶ」ことで取得する。

※リクエストメッセージの例

```
<transport xmlns="http://gutp.jp/fiap-mgmt/2009/11/">
 <header>
  <lookup  … type="component">
   <key id="http://gutp.jp/pointB" …
            gteq="2012-01-01T00:00:00+09:00"
            lt="2012-02-01T00:00:00+09:00" />
  </lookup>
 </header>
</transport>
```

http://gutp.jp/pointB
の2012年1月のデータ
を担当している
IEEE1888コンポーネント
を見つけ出す
(結果は一つとは限らない)

LOOKUP手順 (レスポンス)

```
<transport xmlns="http://gutp.jp/fiap-mgmt/2009/11/">
 <header>
  <lookup  … type="component">
   <key id="http://gutp.jp/pointB" …  />
  </lookup>
  <OK />
 </header>
 <body>
  <component name="main-storage1" … priority="20" uri="…">
    <key id="http://gutp.jp/pointB" … gteq="2012-01-01…"/>
  </component>
  <component name="main-storage2" … priority="20" uri="…">
    <key id="http://gutp.jp/pointB" …  lt="2012-01-01…"/>
  </component>
  <component name="sub-storage" … priority="10" uri="…">
    <key id="http://gutp.jp/pointB" …  />
  </component>
 </body>
</transport>
```

2012年以降
のデータはこちら

2012年より前
のデータはこちら

上記のサーバに
失敗したら、
こちら(バックアップ)

レジストリの役割: その2
各ポイントに付随する情報を管理する

復習 : FETCH, WRITE, TRAPでやり取りされるデータは

```
<point id="http://gutp.jp/pointA">
 <value time="2012-01-01T00:00:00+09:00">25.5</value>
 <value time="2012-01-01T00:30:00+09:00">25.6</value>
 <value time="2012-01-01T01:00:00+09:00">26.7</value>
 …
</point>
```

のようなフォーマットを持つ

ただし、これだけだと、http://gutp.jp/pointA が何なのかは、
明示的にはわからない

「http://gutp.jp/pointA　は、温度(℃)であり、精度は0.1ステップ、30分単位で計測する」
のような定義情報が必要になる
➔ 通常はポイント表で管理する (本日午後のテーマ)が、レジストリを使えば動的に管理できる

ポイントに付随する情報の管理

どのような方法で行うか？

1. 属性空間を定義する
   - XML名前空間: http://example.org/mySensorSetting
   - devtype属性: temp (温度℃センサ), hum(相対湿度%センサ)
   - step属性: 計測の精度 (1 または 0.1)
   - period属性: 計測の周期 {1, 5, 10, 30, 60} のどれか
2. ポイントに情報を並べる

```
xmlns:s=http://example.org/mySensorSetting
<point id="http://gutp.jp/pointA" s:devtype="temp" s:step="0.1" period="30"/>
<point id="http://gutp.jp/pointB" s:devtype="hum" s:step="1" period="30"/>
```

⇨ これらをレジストリに登録したり、検索したりする。

ポイント情報の管理で可能になること:その1

外部団体が決めているデータモデルを参照できるようになる

IEEE1888ポイントID

http://gutp.jp/pointA echonet:class="21" echonet:instance="2"
http://gutp.jp/pointB echonet:class="23" echonet:instance="3"

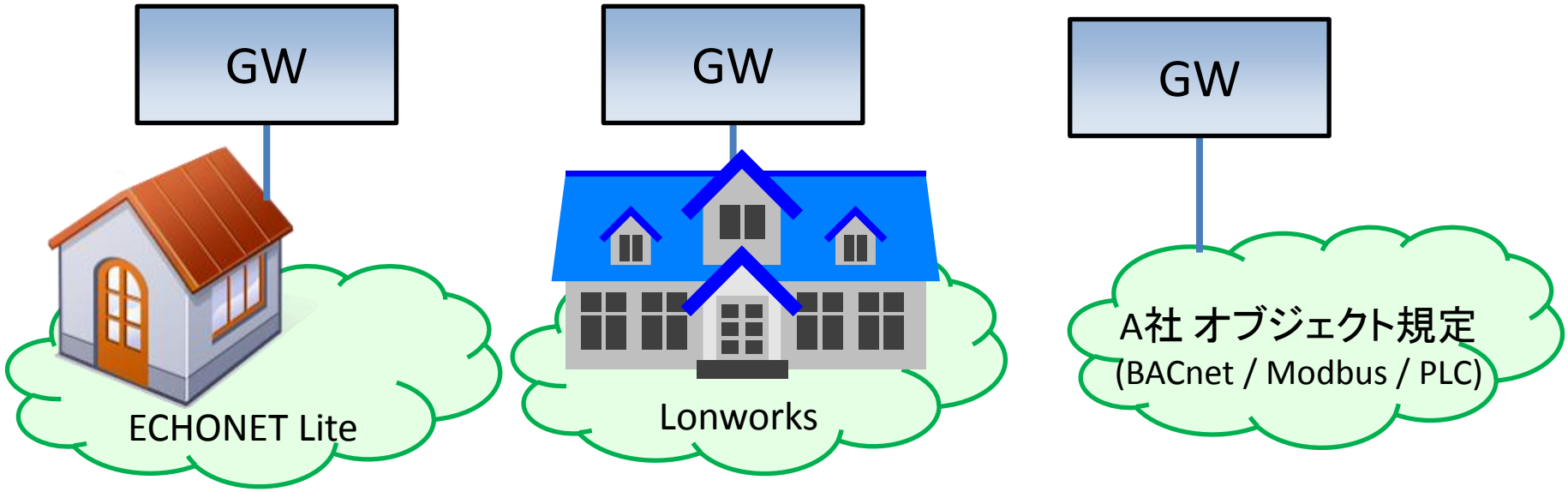
} ECHONETのどのようなデータであるかが判明する

http://gutp.jp/pointC lon:snvt="SNVT_TEMP_P"
http://gutp.jp/pointD lon:snvt="SNVT_switch"

} Lonworksのどのようなデータであるかが判明する

http://gutp.jp/pointE a:plc_address="40001"
http://gutp.jp/pointF a:plc_address="40003"

} A社システムにおけるどのようなデータであるがが判明する



それぞれの通信規格で、独自の機器モデル体系を持っている

ポイント情報の管理で可能になること:その2

システム構成の変化に、ダイナミック対応できるようになる

IEEE1888ポイントID

http://gutp.jp/pointA　deploy:room="102"　deploy:status="active"
http://gutp.jp/pointB　deploy:room="102"　deploy:status="active"
http://gutp.jp/pointC　deploy:room="103"　deploy:status="active"
http://gutp.jp/pointD　deploy:room="103"　deploy:status="inactive"

active: 運用中
inactive: 過去に運用したが
現在は停止中

新規にポイントを追加する
http://gutp.jp/pointE　deploy:room="102"　deploy:status="active"

部屋"102" のセンサー覧表に
　http://gutp.jp/pointA, http://gutp.jp/pointB　に加えて、
　http://gutp.jp/pointE　が (いつの間にか)表示されるようになる
属性を deploy:status="inactive" にすれば、そのセンサが一覧画面から消える

本日(16日)のスケジュール

10:00　IEEE1888レジストリの仕組み

10:30　レジストリを用いたCEMS実現への取組み

11:00　IEEE1888レジストリとの通信プログラミング実習

12:00　昼食

13:00　IEEE1888システムの設計と構築：その1

14:30　休憩

15:00　IEEE1888システムの設計と構築：その2

16:00　簡易な相互接続実験

17:00　解散

本日(16日)のスケジュール

10:00　IEEE1888レジストリの仕組み

10:30　レジストリを用いたCEMS実現への取組み

11:00　IEEE1888レジストリとの通信プログラミング実習

12:00　昼食

13:00　IEEE1888システムの設計と構築：その１

14:30　休憩

15:00　IEEE1888システムの設計と構築：その２

16:00　簡易な相互接続実験

17:00　解散

レジストリ・プログラミング

- レジストリのサンプルコード＆マニュアル
  - http://133.11.168.3/FIAPRegistryClient.zip

IEEE1888 Registry プログラミング・マニュアル
～クライアント編～

作成日: 2012 年 2 月 23 日
最終更新: 2012 年 10 月 16 日
作成者: 落合秀也(東京大学)

1．概要

本マニュアルは、IEEE1888 コンポーネント対レジストリ間通信のクライアント側のプログラミング方法を、記載しています。レジストリとの通信においては、“コンポーネントの登録&検索”と“ポイントの登録&検索”があるが、前者のプログラミング方法のみを解説します。

Lookupのサンプルコード

```
<?php

// UUIDの生成関数
function uuid(){
  return sprintf('%04x%04x-%04x-%04x-%04x-%04x%04x%04x',
    mt_rand( 0, 0xffff ), mt_rand( 0, 0xffff), mt_rand( 0, 0xffff ),
    mt_rand( 0, 0x0fff ) | 0x4000,
    mt_rand( 0, 0x3fff ) | 0x8000,
    mt_rand( 0, 0xffff ), mt_rand( 0, 0xffff), mt_rand( 0, 0xffff ));
}

// RegistryサーバのWSDL
$wsdl="http://133.11.168.3/axis2/services/FIAPMgmtWS?wsdl";

// 検索するポイントIDの設定
$key1=array("id"=>"http://www.gutp.jp/dummy/integer",  // 書き換えてください
          "attrName"=>"time");                    // 例: http://gutp.jp/interop01/Temperature

// リクエストのオブジェクト生成
$lookup=array("type"=>"component","id"=>uuid(),"key"=>$key1);
$header=array("lookup"=>$lookup);
$transport=array("header"=>$header);
$lookupRQ=array("transport"=>$transport);

$server = new SoapClient($wsdl);

// リクエスト内容(オブジェクト構造)の表示
var_dump($lookupRQ);

$lookupRS = $server->lookup($lookupRQ);  // lookupリクエストの発行

// 応答内容(オブジェクト構造)の表示
var_dump($lookupRS);

?>
```

実践: レジストリ・プログラミング

- SDK内で、以下のサンプルコードを動かし、動作を確認します。
  - registration.php (Storageサーバを登録する)
    (*) 適切な個所($name= や $uri=)を、
    そのSDKのStorageを表すように編集してください
  - lookup.php (他者のサーバを発見する)
    (*) 問い合わせる内容を、いろいろ変えてみてください

- なお、SDKは会場ネットワーク(192.168.10.x/24)に接続してください

本日(16日)のスケジュール

10:00　IEEE1888レジストリの仕組み

10:30　レジストリを用いたCEMS実現への取組み

11:00　IEEE1888レジストリとの通信プログラミング実習

12:00　昼食

13:00　IEEE1888システムの設計と構築：その１

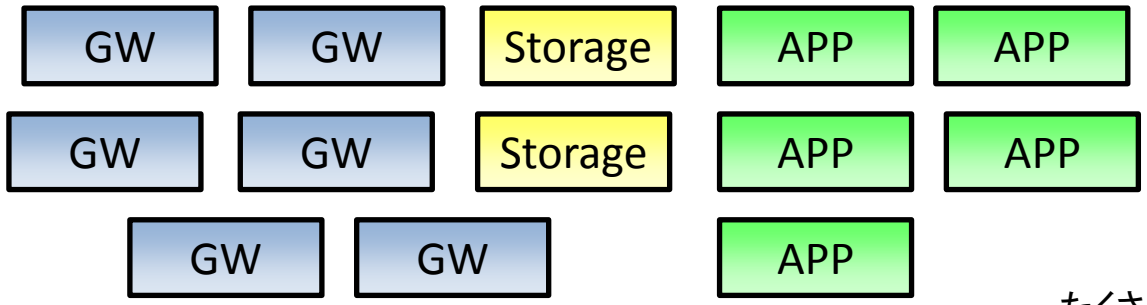
14:30　休憩

15:00　IEEE1888システムの設計と構築：その２

16:00　簡易な相互接続実験

17:00　解散

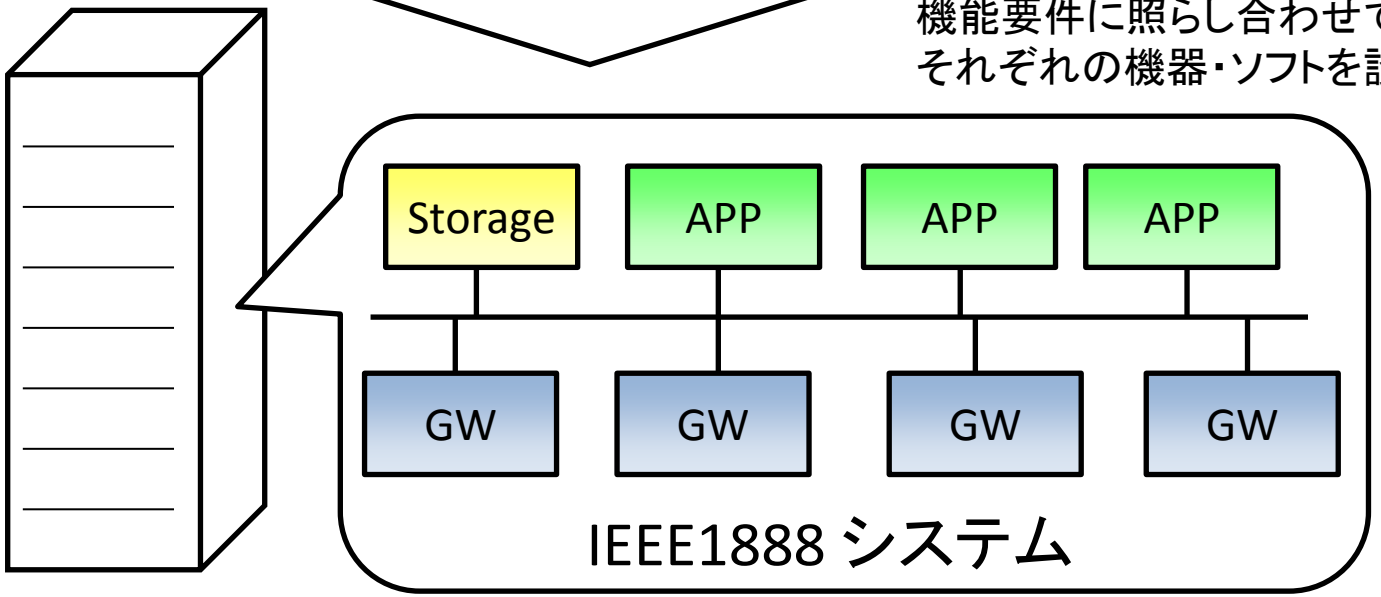
システム設計の心構え: 基本的な仕組み



IEEE1888部品市場

たくさんの種類の GW, Storage, APPが入手できる

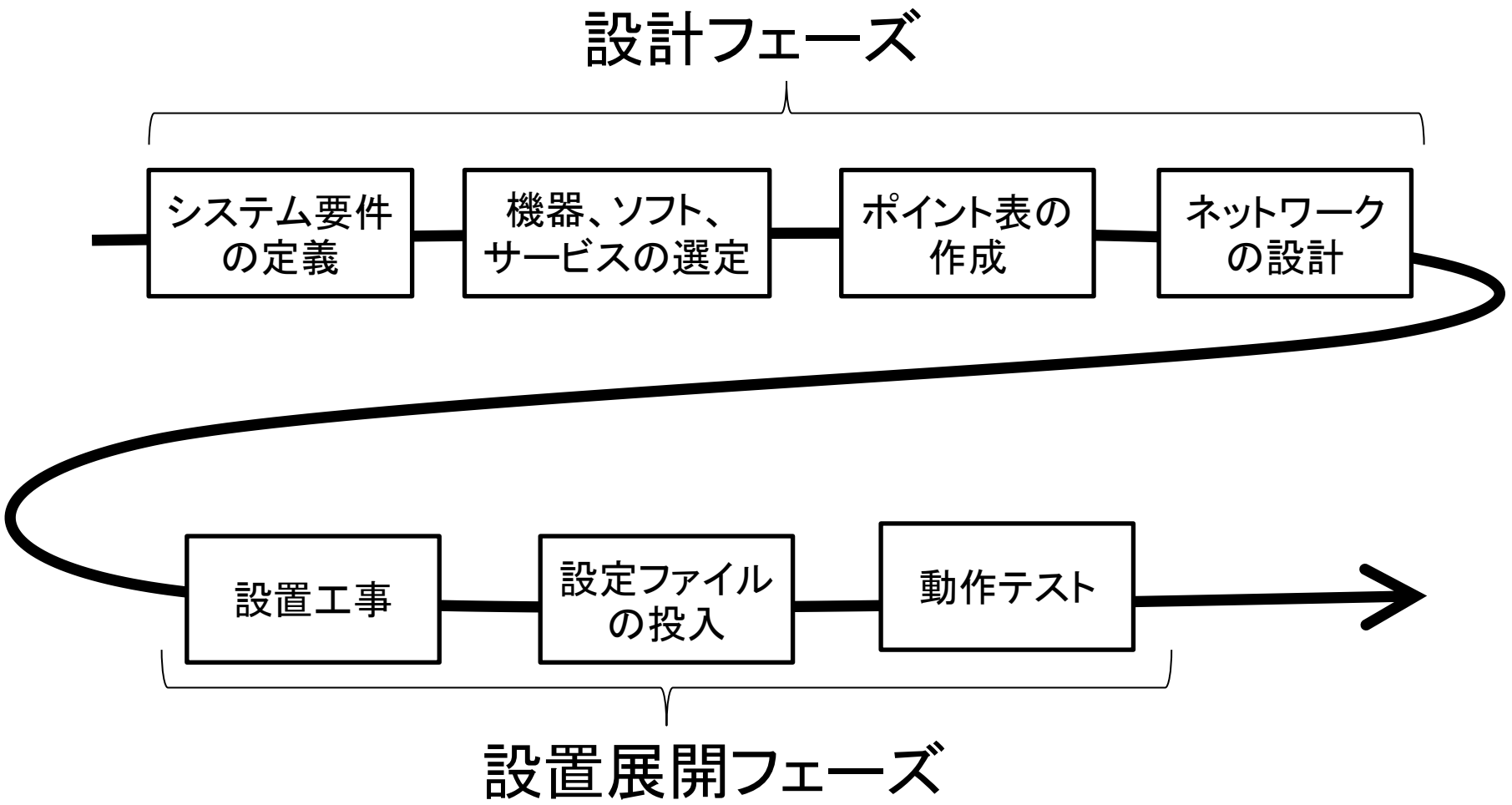
機能要件に照らし合わせて必要なものを選定＆購入
それぞれの機器・ソフトを設定して連携させる



IEEE1888 システムの実現

導入現場

工程 (設計･展開) の概観

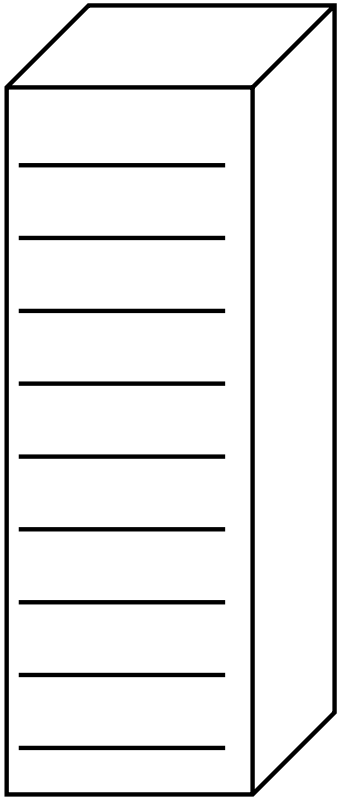

設計フェーズ
システム要件の定義
機器、ソフト、サービスの選定
ポイント表の作成
ネットワークの設計
設置工事
設定ファイルの投入
動作テスト
設置展開フェーズ

ステップ1：要件定義

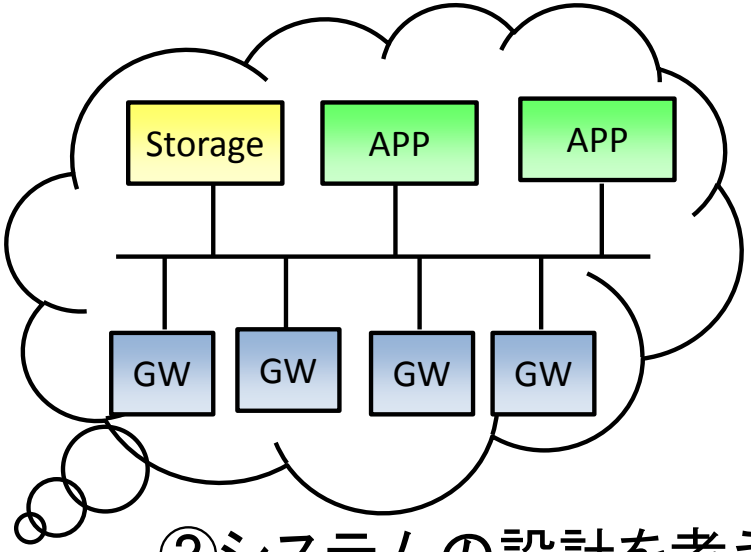
項目	検討の観点
検討1	どのようなユーザーインタフェースを持たせるか？
検討2	どのような制御を働かせるか？
検討3	何を計測しなければならないか？
検討4	何を制御できなればならないか？
検討5	どのような計算機能や演算ロジックが必要になるか？
検討6	どのくらいの頻度で，どの期間のデータを蓄積する必要があるか？
検討7	どの程度のサービス可用性を求めるか？
検討8	どのくらいの規模までを扱うか？
検討9	その他，現場特有の要件は？
検討10	今回の工事で対象としないことは何か？

この検討項目は一例。実際には現場に応じて、追加検討しなければいけない項目も出てくる。

ステップ2：
機器・ソフトウェア・サービスの選定



導入現場



①調査
or 想定

②システムの設計を考える

設備ネットワーク
システムエンジニア

③必要な機器・ソフトウェア・サービスの選定

- 計測にはA社のZigBee電力メータを使おう
- その電力メータ用のB社のIEEE1888 ZigBee GWを使おう
- IEEE1888 Storage はC社運用のサービスを使おう
- 見える化にはD社のIEEE1888 簡易SCADAソフトを使おう
- メール発行にはF社の IEEE1888監視メーラーを使おう

ステップ3：ポイント表の設計

ポイントID	場所	名称	単位	その他
http://gutp.jp/Room1/PE	部屋1	積算電力量	kWh	100000 で 0クリア 0.001ステップで増加 1分おきに更新
http://gutp.jp/Room1/Demand	部屋1	予測デマンド値	kW	(注1)
http://gutp.jp/Room1/TargetDemand	部屋1	目標デマンド値	kW	(注2)
http://gutp.jp/Room1/Temp	部屋1	温度	℃	0.1ステップ 計測範囲: 0℃～40℃ ±2℃程度の観測漏れあり
http://gutp.jp/Room1/Hum	部屋1	相対湿度	%	1ステップ 計測範囲: 10%～90%
http://gutp.jp/Room1/CO2	部屋1	CO2濃度	ppm	10ステップ 計測範囲: 200ppm ～ 3000ppm
http://gutp.jp/Room1/LightStat	部屋1	照明ONOFF状態	{true, false}	true: ON状態; false: OFF状態
http://gutp.jp/Room1/LightCtrl	部屋1	照明ONOFF制御	{true, false}	true: ON状態; false: OFF状態
….	…	…		

(注1) 積算電力量の変化から予測されるデマンド時限での予測量 (更新周期は・・・)
(注2) 目標デマンド量を変化させたいときに更新する (デマコンへの入力)

ポイントを定義し、その値の意味をできるだけ詳細に渡って設計する。

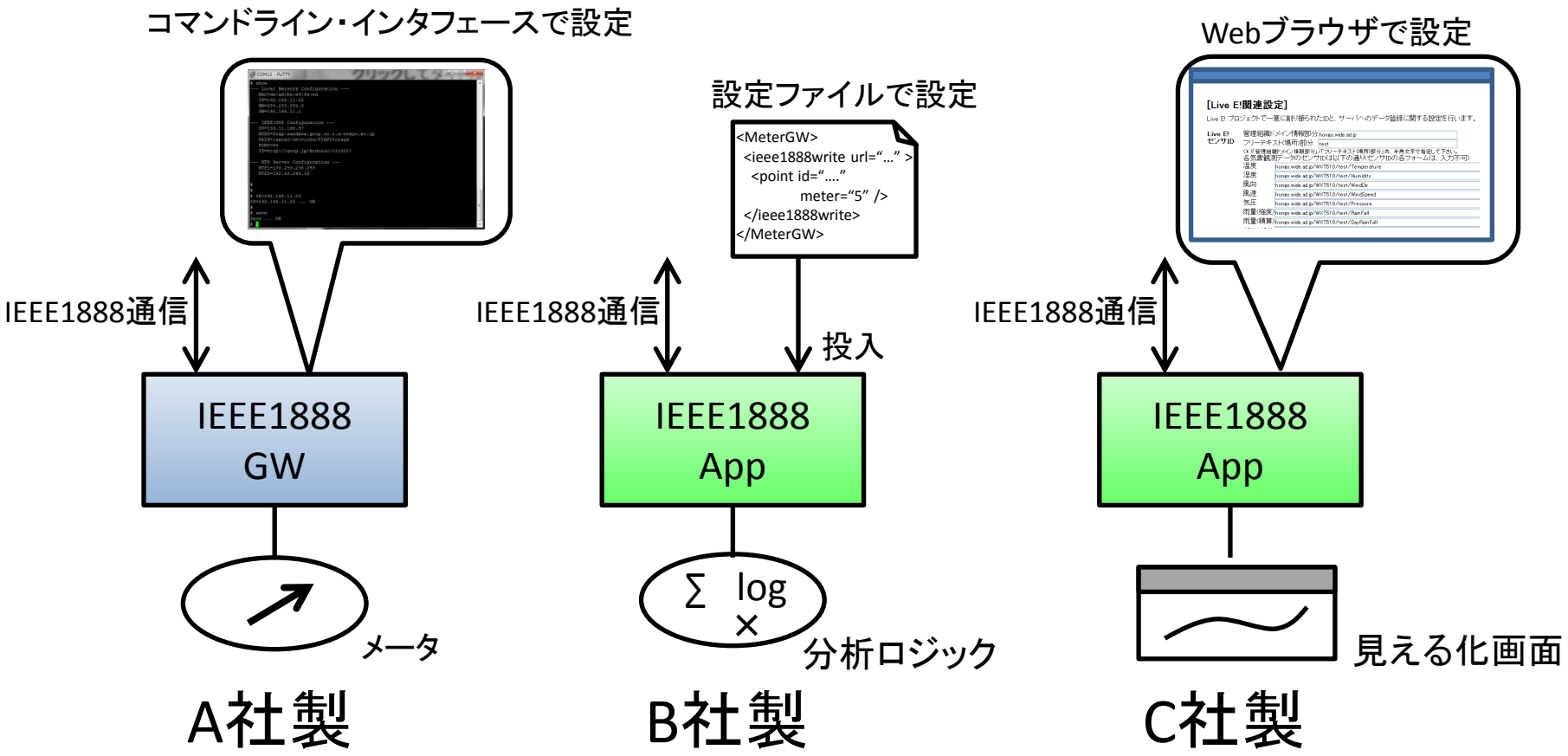
ステップ4：IPネットワークの設計

機器・ソフト・サービス	ホスト名	IPv4アドレス	IPv6アドレス	URL
IEEE1888スマートタップGW	smarttap-gw	133.11.168.85	2001:200:180:8011::85	…
IEEE1888人感・照度センサGW	md-gw	133.11.168.86	2001:200:180:8011::86	…
IEEE1888 Storage(1)	storage1	133.11.168.87	2001:200:180:8011::87	…
IEEE1888 Storage(2)	storage2	133.11.168.88	2001:200:180:8011::88	…
IEEE1888テスト用Storage	fiap-develop	133.11.168.39	2001:200:180:8011::39	…
IEEE1888積算電力解析APP	fiap-dev	133.11.168.37	2001:200:180:8011::37	…
IEEE1888簡易プロトコルテスター	fiap-dev	133.11.168.37	2001:200:180:8011::37	…
IEEE1888電力メータGW	power-meter-gw	133.11.168.90	2001:200:180:8011::90	…
IEEE1888数値表示APP	number-display-app	133.11.168.91	2001:200:180:8011::91	…

どの機器に、どのホスト名、IPアドレスを割り当てるかを設計する。
サーバの場合は、通信のURLも記載する。

ステップ5, 6：設置＆設定の投入

様々な種類の設定用のインタフェースが存在する



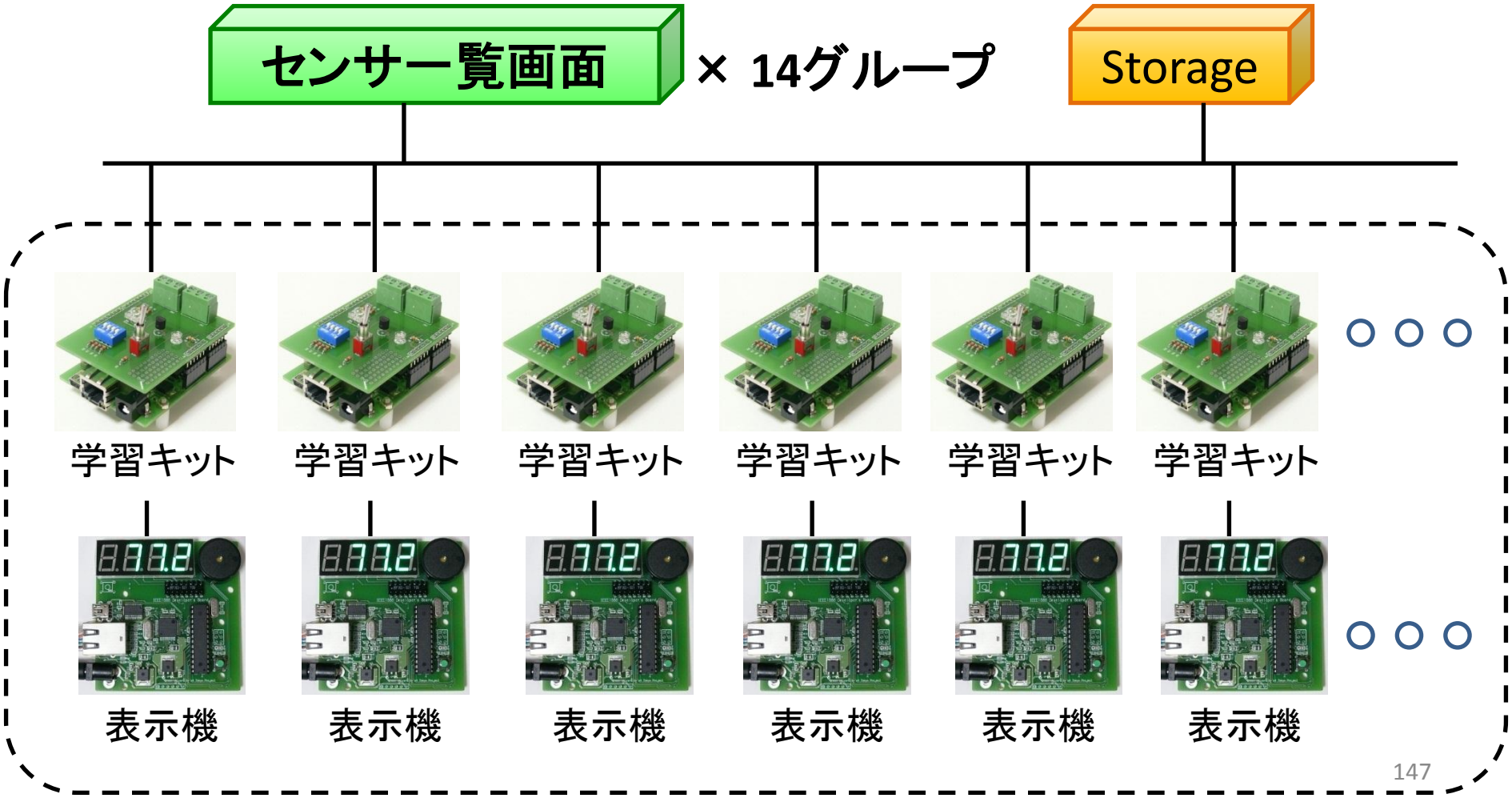
ステップ7：動作の検証

- 設定したポイントが正しい値を出しているかの確認
  - 良くある工事ミス
    - 紐付ミス (部屋番号の間違い)
    - 倍率設定ミス (10倍の値が出力されていた)
- 全体のシステム要件を満足するかのテスト
  - 要件毎に、満足するかどうかを確認する

本実践ワークショップで

今から、設計＆構築するシステム

温度および照度の管理・警報システムを作りましょう



システム要件の設定 (1/3)

- 検討1: どのようなユーザーインタフェースを持たせるか？
  - 計測対象、温度、照度が一覧で表示される
  - 温度、照度の履歴を、それぞれグラフで確認できる
  - CSVファイルに出力することができる

- 検討2:どのような制御を働かせるか？
  - 近隣の温度をLEDディスプレイに(0.1ステップで)表示
  - 30℃、32℃、33℃でレベル1、2、3の警告音を発生 (温度が高いと警告)
  - 制御の遅れは1分程度は許容する

システム要件の設定 (2/3)

- 検討3：何を計測しなければならないか？
  - 温度、照度
- 検討4：何を制御しなければならないか？
  - LEDディスプレイ（数値の表示機）
  - 警告ブザー
- 検討5：どのような計算機能や演算ロジックが必要になるか？
  - 特になし

システム要件の設定 (3/3)

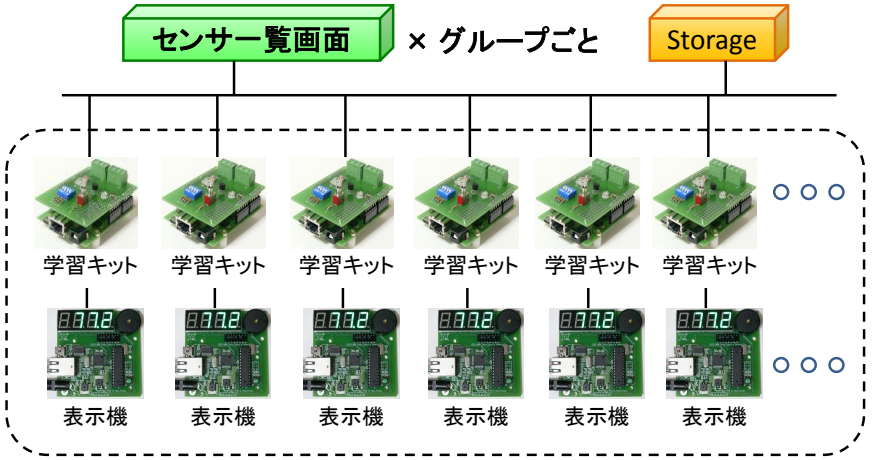
- 検討6：どのくらいの頻度で，どの期間のデータを蓄積する必要があるか？
  - 1分に1回の頻度で、ワークショップ期間中保管する。
    （実際の例）: 1分に1回の頻度で、2年間は保管する。
- 検討7：どの程度のサービス可用性を求めるか？
  - ワークショップの期間中、稼働すればよい
    （実際の例）: 年3回（1回あたり12時間）メンテナンスにより停止する。故障等による停止の場合は、1週間以内に復旧。
- 検討8：どのくらいの規模までを扱うか？
  - 15ヵ所の計測
  - 10ユーザ/秒の閲覧リクエスト

機器・ソフトウェア・サービスの選定(1/3)

- 今回の要件の下では、
  - 温度・照度の計測器 (GW)
  - LEDディスプレイ表示機、ブザーによる警報機 (GW)
  - データ蓄積装置 (Storage)
  - センサー覧管理画面 (APP)

  を選定する必要がある。

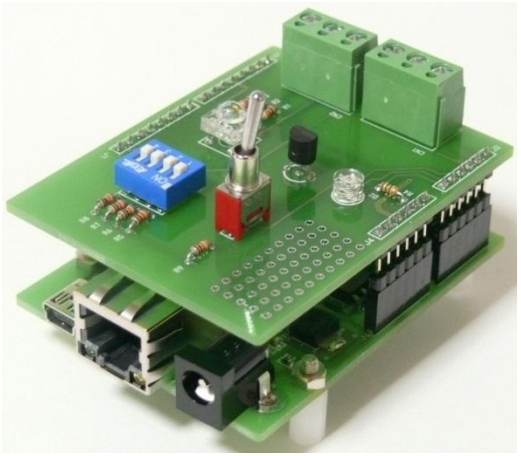
➔ これらをうまく組み合わせれば、システムを構築できる。



機器・ソフトウェア・サービスの選定(2/3)

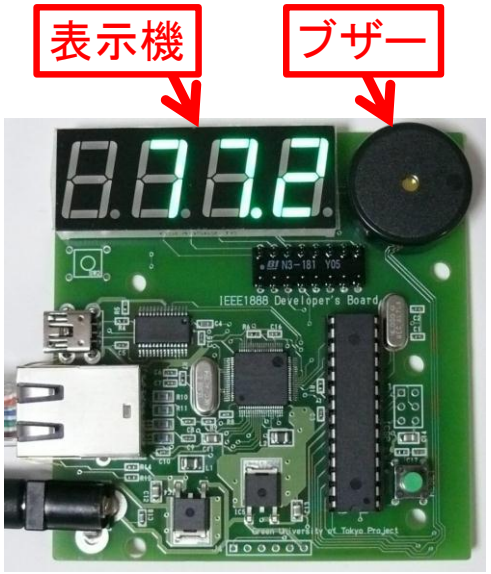
- 温度・照度の計測器 (GW)

IEEE1888開発ボード(学習キット)
が使える



- LEDディスプレイ表示機
  ブザーによる警報機 (GW)

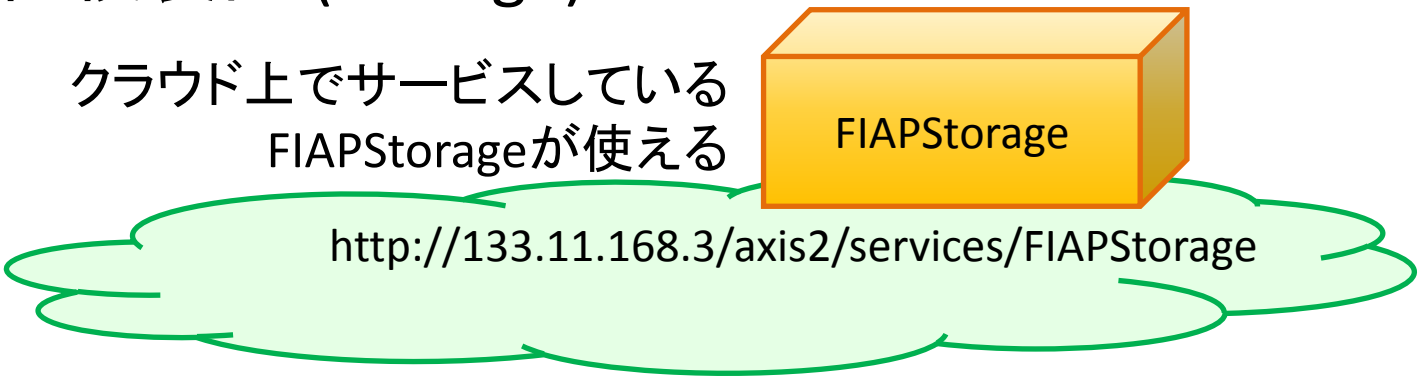
ピークカットヘルパー
が使える



機器・ソフトウェア・サービスの選定(3/3)

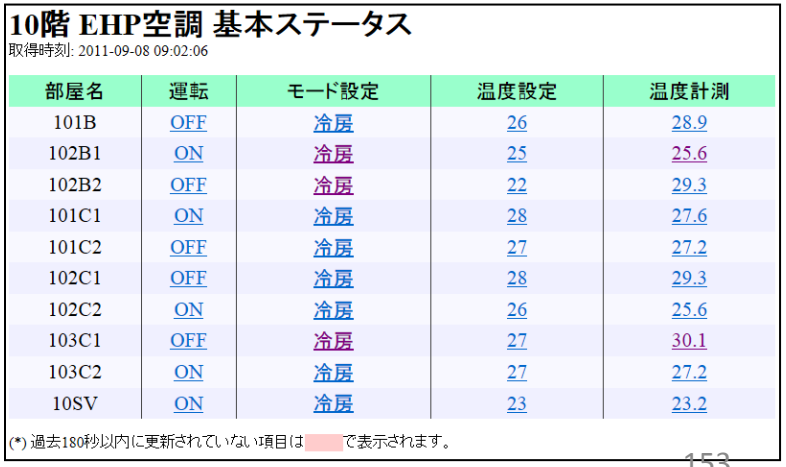
- データ蓄積装置 (Storage)

クラウド上でサービスしている
FIAPStorageが使える



- センサー覧管理画面 (APP)

FIAPSimpleSCADAという
簡易Web画面ツール
が使える



10階 EHP空調 基本ステータス

取得時刻: 2011-09-08 09:02:06

部屋名	運転	モード設定	温度設定	温度計測
101B	OFF	冷房	26	28.9
102B1	ON	冷房	25	25.6
102B2	OFF	冷房	22	29.3
101C1	ON	冷房	28	27.6
101C2	OFF	冷房	27	27.2
102C1	OFF	冷房	28	29.3
102C2	ON	冷房	26	25.6
103C1	OFF	冷房	27	30.1
103C2	ON	冷房	27	27.2
10SV	ON	冷房	23	23.2

(*) 過去180秒以内に更新されていない項目は　で表示されます。

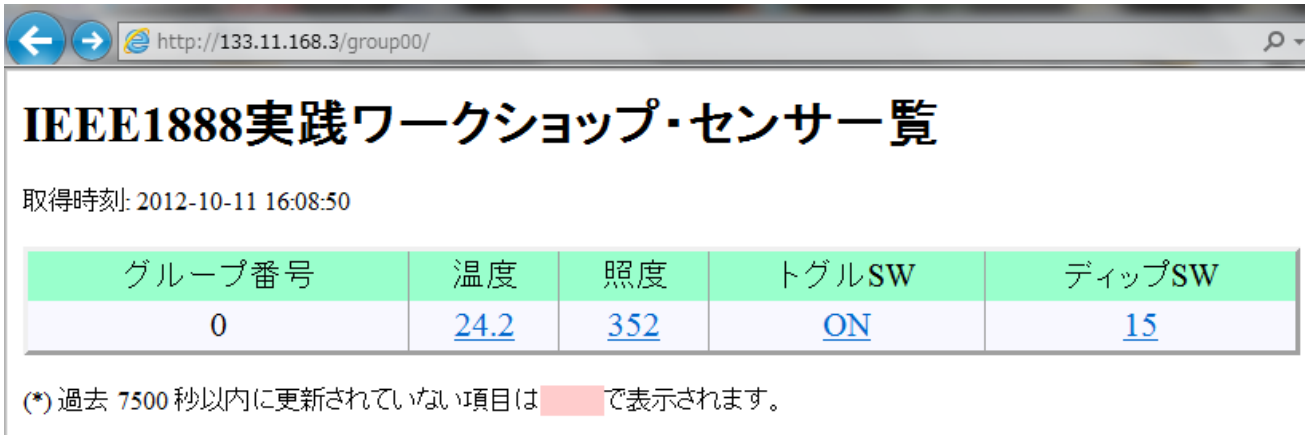
FIAPSimpleSCADA

FIAPSimpleSCADA の例

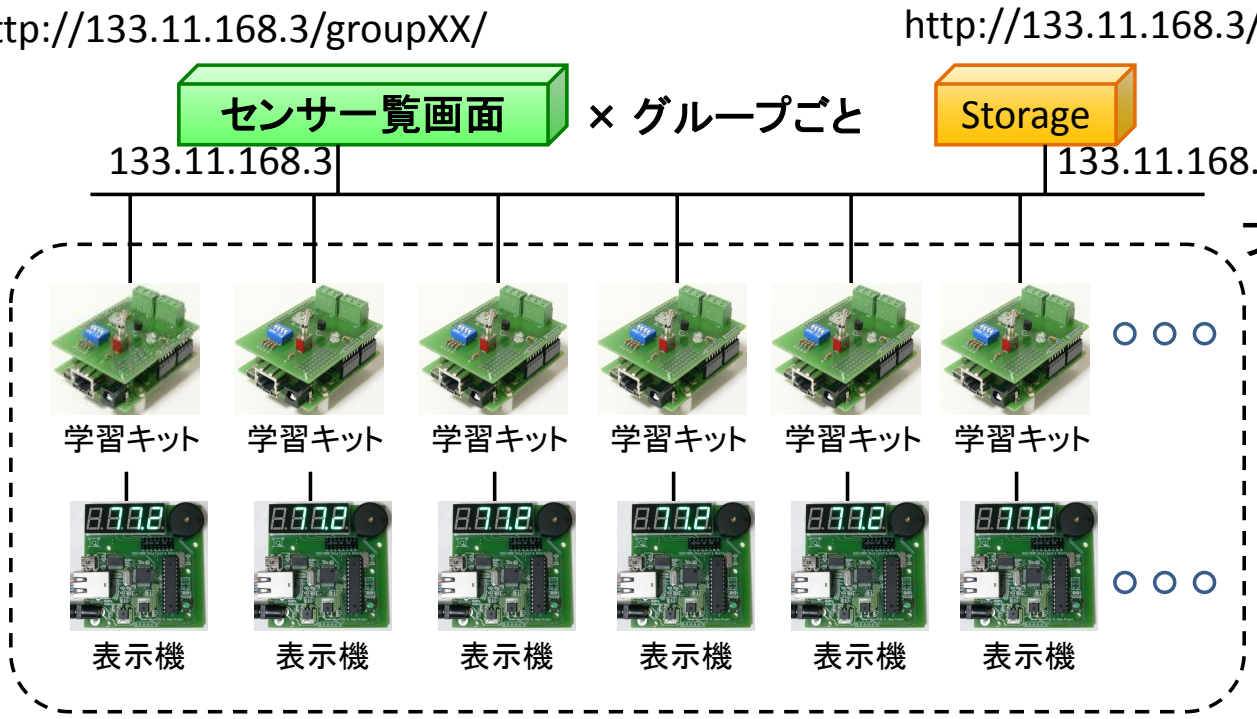
```
<?xml version="1.0" encoding="UTF-8"?>
- <fiapSimpleSCADA title="IEEE1888実践ワークショップ・センサー一覧" fiapURI="http://localhost/a
xmlns="http://gutp.jp/fiap/fiapSimpleSCADA/2011/06/">
  - <entityGroup>
    - <entity name="センサー一覧">
      - <table title="IEEE1888実践ワークショップ・センサー一覧" validityTime="7500">
        - <row>
            <col value="グループ番号"/>
            <col value="温度"/>
            <col value="照度"/>
            <col value="トグルSW"/>
            <col value="ディップSW"/>
          </row>
            <!-- Group 00 -->
        - <row>
            <col value="0"/>
          - <col type="numeric" id="http://gutp.jp/interop00/Temperature" rou
              <trend title="Group0 温度" to="0" from="-3600"/>
            </col>
          - <col type="numeric" id="http://gutp.jp/interop00/Illuminance" rou
              <trend title="Group0 照度" to="0" from="-3600"/>
            </col>
          - <col type="string" id="http://gutp.jp/interop00/TGLSW">
              <trend title="Group0 TGLSW 状態" to="0" from="-3600"/>
            </col>
          - <col type="string" id="http://gutp.jp/interop00/DIPSW">
              <trend title="Group0 DIPSW 状態" to="0" from="-3600"/>
            </col>
          </row>
```

設定内容

画面構成



実践: 実際に設計してみよう



プライベートネットワーク
192.168.10.0/24

各グループのIPアドレス
192.168.10.X0
(Xはグループ番号)

※ その他の情報
GW: 192.168.10.254
DNS: 192.168.10.254
NTP: 133.243.238.243

ステップ3: ポイント表の設計

ステップ4: ネットワーク設計

本日(16日)のスケジュール

10:00　IEEE1888レジストリの仕組み

10:30　レジストリを用いたCEMS実現への取組み

11:00　IEEE1888レジストリとの通信プログラミング実習

12:00　昼食

13:00　IEEE1888システムの設計と構築: その1

14:30　休憩

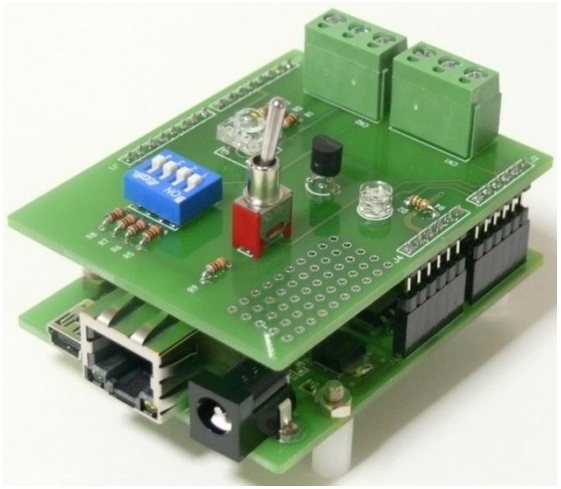
15:00　IEEE1888システムの設計と構築: その2

16:00　簡易な相互接続実験

17:00　解散

IEEE1888システムの設計と構築：その2
設計したシステムを実現する

- 設計に基づいて、実際に機器(＆サービス)に設定を入れます。



IEEE1888学習キット

ピークカットヘルパー

10階 EHP空調 基本ステータス

取得時刻: 2011-09-08 09:02:06

部屋名	運転	モード設定	温度設定	温度計測
101B	OFF	冷房	26	28.9
102B1	ON	冷房	25	25.6
102B2	OFF	冷房	22	29.3
101C1	ON	冷房	28	27.6
101C2	OFF	冷房	27	27.2
102C1	OFF	冷房	28	29.3
102C2	ON	冷房	26	25.6
103C1	OFF	冷房	27	30.1
103C2	ON	冷房	27	27.2
10SV	ON	冷房	23	23.2

(*) 過去180秒以内に更新されていない項目は で表示されます。

FIAPSimpleSCADA

昨日と同様の手順 (コマンドライン) で設定

サーバにログインして設定

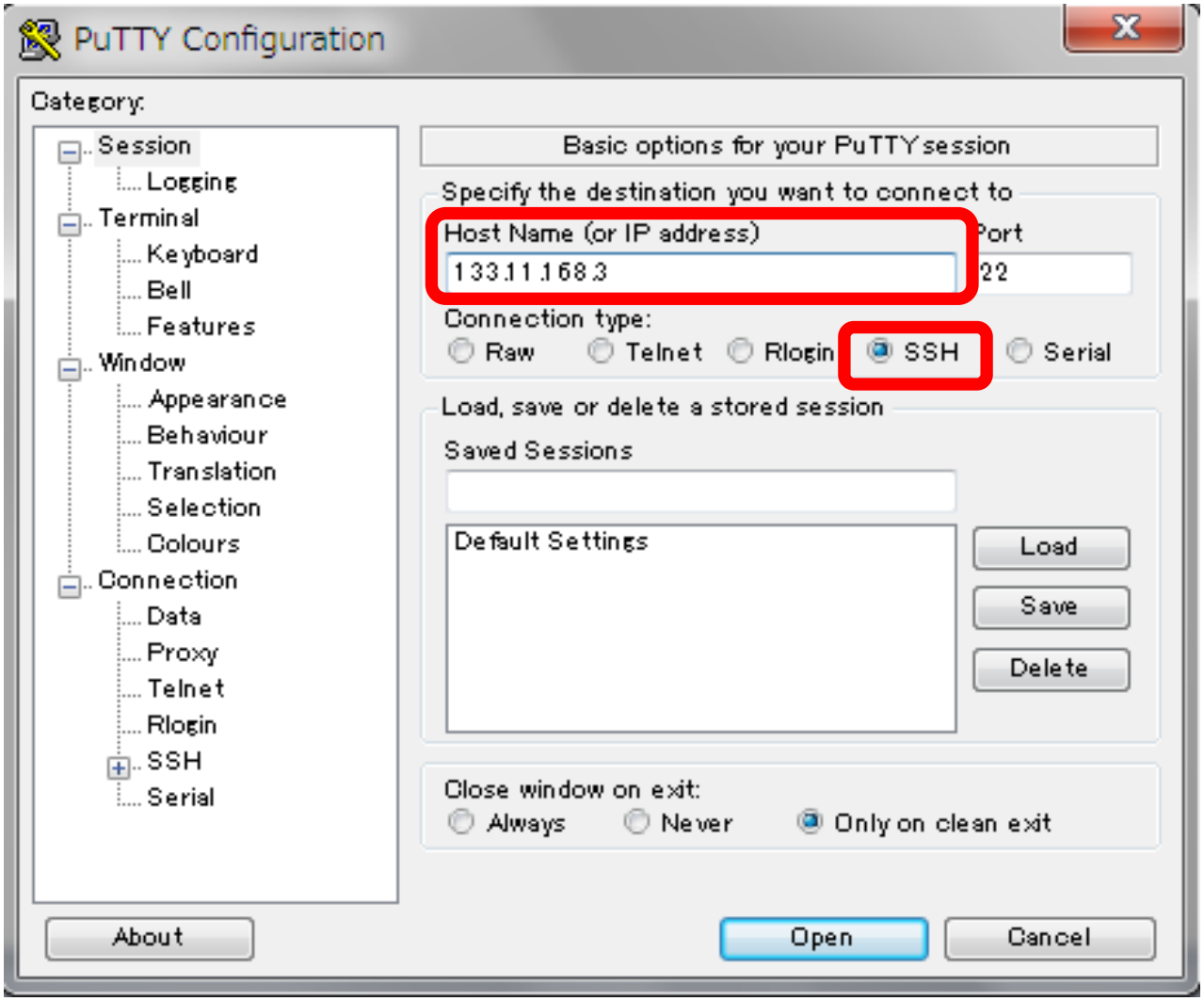
IEEE1888システムの設計と構築：その２
各グループに（用意された）FIAPSimpleSCADA

グループ番号	閲覧画面	設定ファイルの場所 (サーバに遠隔ログインしてアクセスする)
1	http://133.11.168.3/group01/	/var/www/group01/fiapSimpleSCADA.xml
2	http://133.11.168.3/group02/	/var/www/group02/fiapSimpleSCADA.xml
3	http://133.11.168.3/group03/	/var/www/group03/fiapSimpleSCADA.xml
4	http://133.11.168.3/group04/	/var/www/group04/fiapSimpleSCADA.xml
5	http://133.11.168.3/group05/	/var/www/group05/fiapSimpleSCADA.xml
…	…	…

自分のグループのFIAPSimpleSCADA
閲覧画面にアクセスしてみましょう。

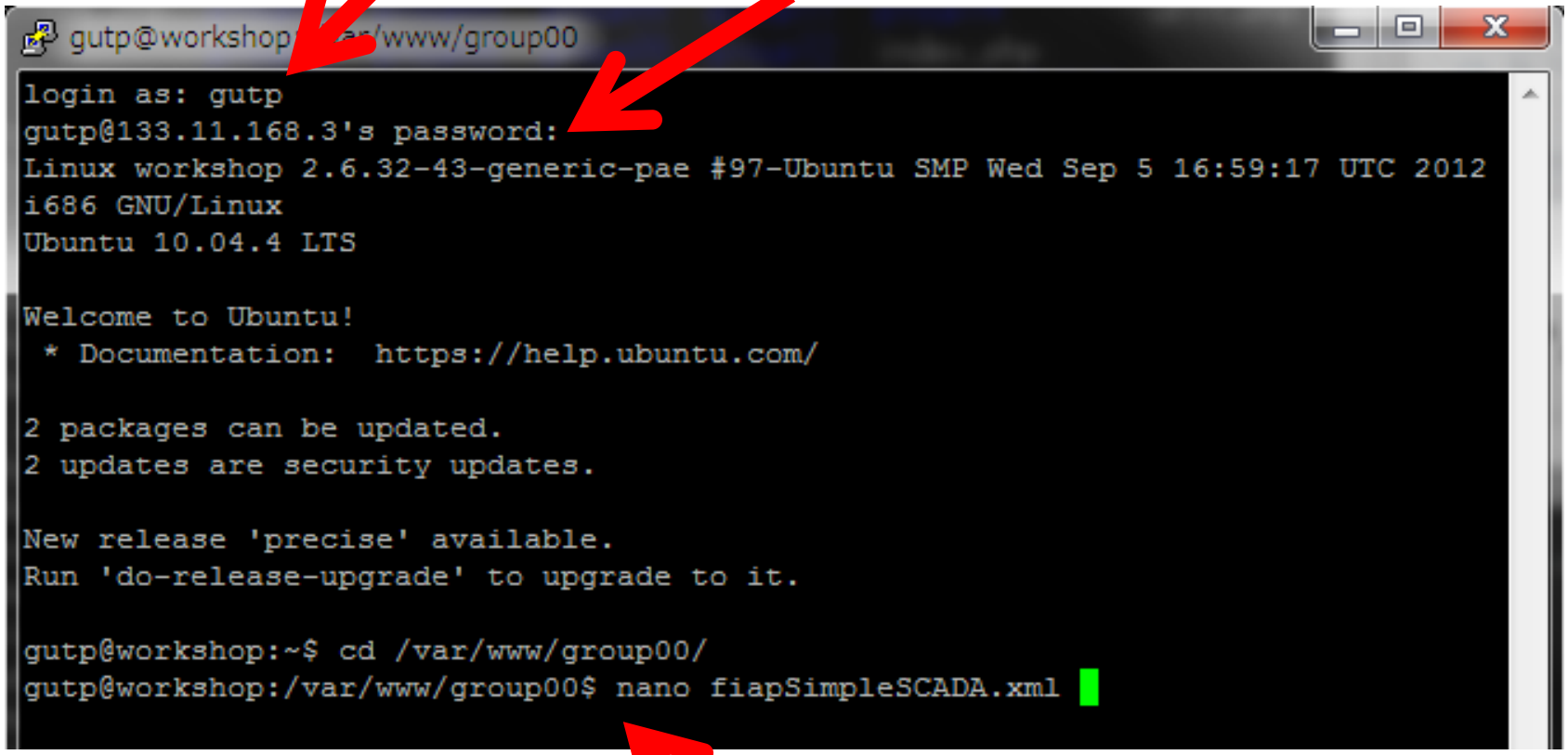
* 遠隔ログイン(SSHログイン)情報
  IPアドレス: 133.11.168.3
  ユーザ名: gutp
  パスワード: ieee1888

FIAPSimpleSCADA設定ファイルの編集 その1：遠隔ログイン方法



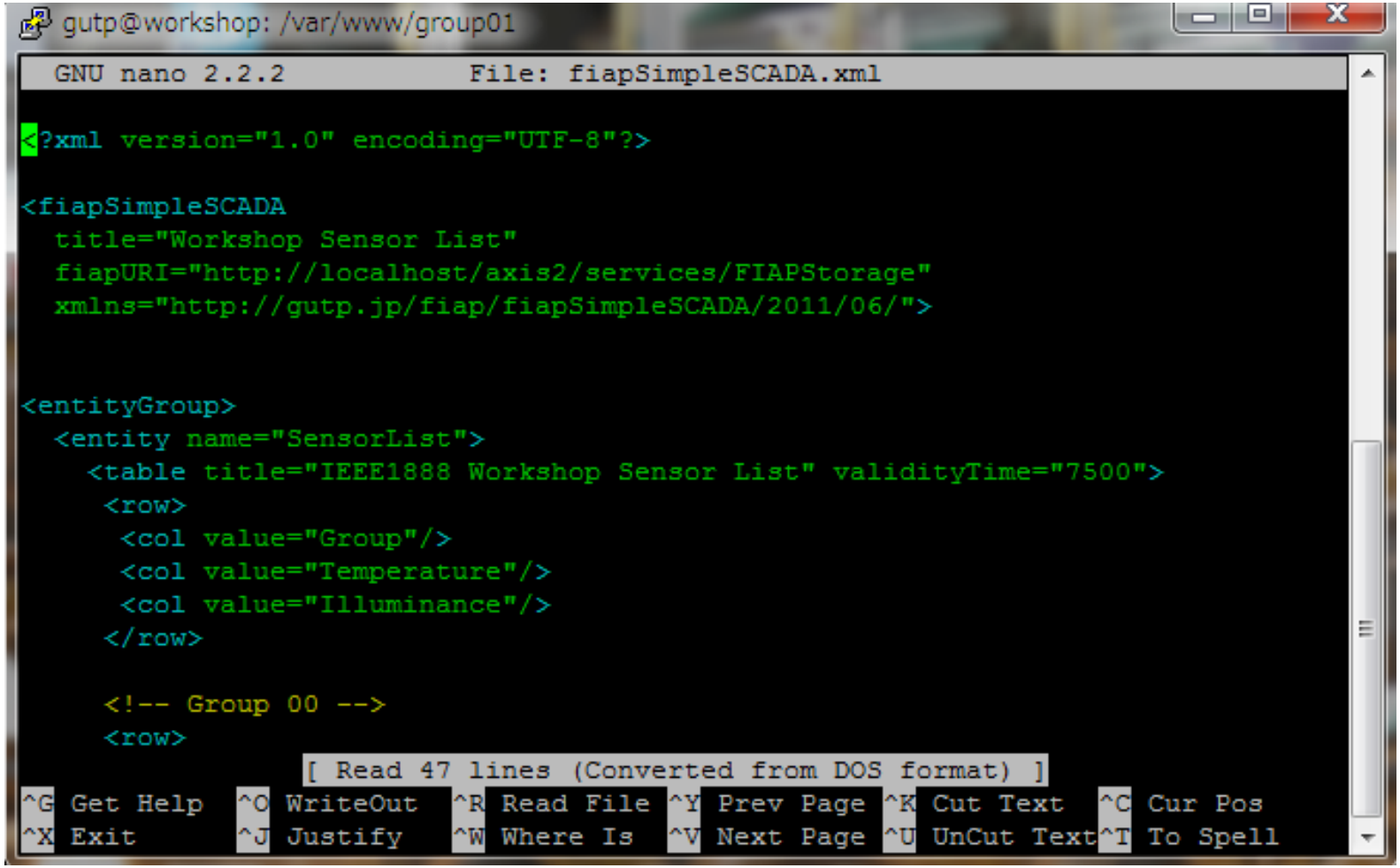
FIAPSimpleSCADA設定ファイルの編集 その2：遠隔ログイン&エディタの起動

gutpと入力　ieee1888と入力



設定ファイルの場所に進み、nanoコマンド(vimでも良い)で設定ファイルを編集する

FIAPSimpleSCADA設定ファイルの編集
その3：nanoエディタを使い設定ファイルを編集

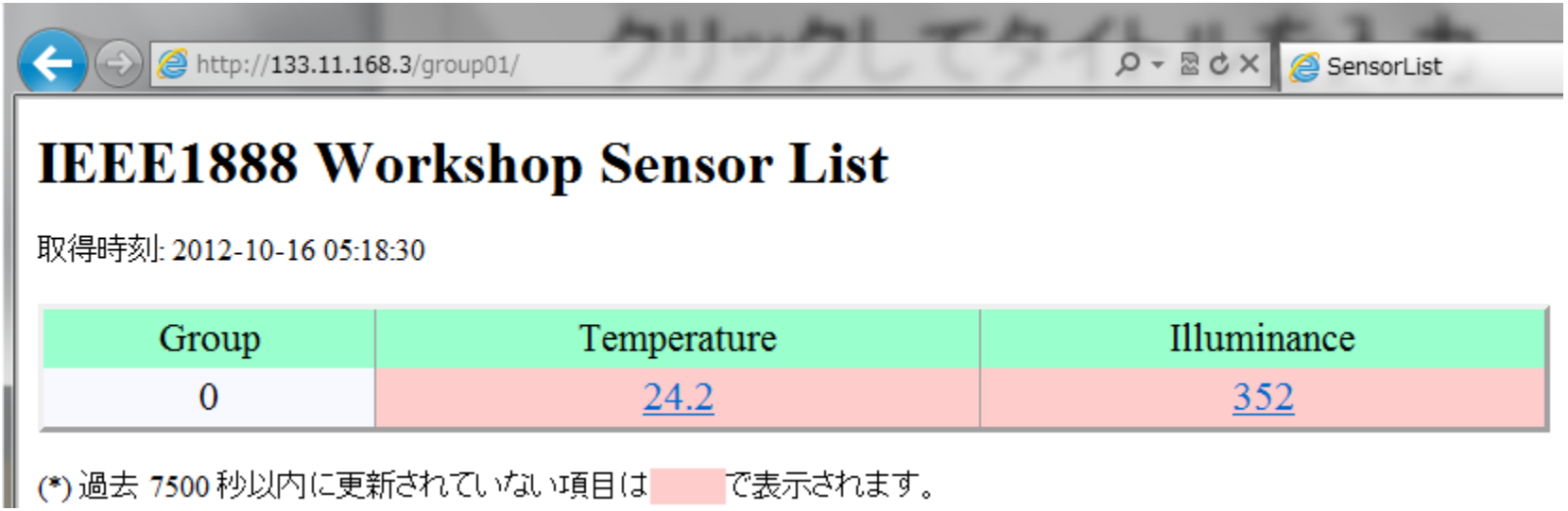


使い方

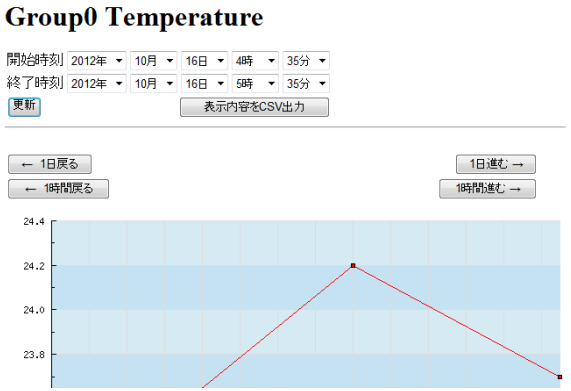
Controlキー + O で「ファイルを保存」(WriteOut)
Controlキー + X で「プログラムの終了」(Exit)

FIAPSimpleSCADA設定ファイルの編集
その４：閲覧用のページで確認する

例：グループ1の場合は http://133.11.168.3/group01/



リンクをクリックすると、
トレンドデータを確認できる。

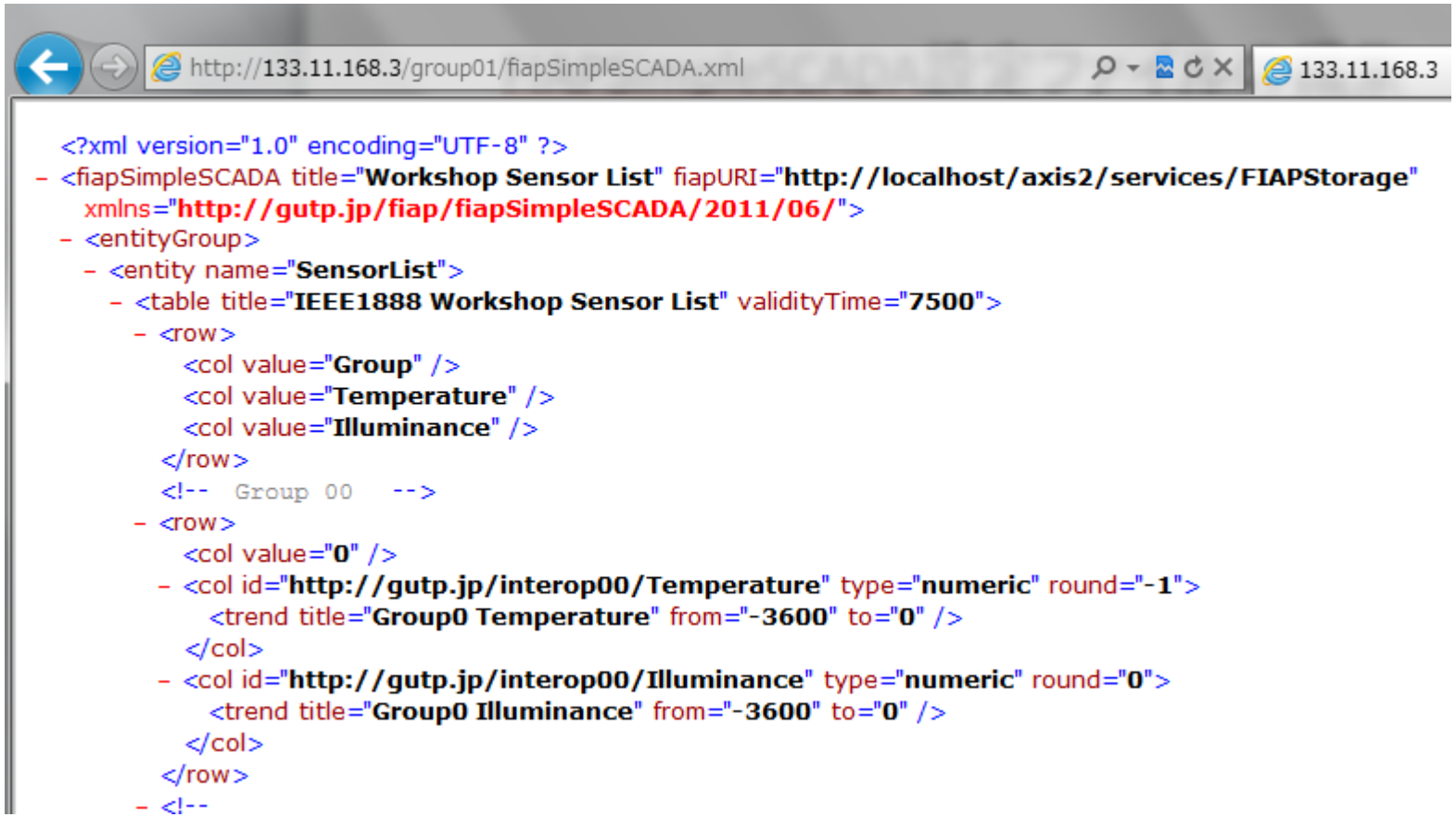


他グループが作った閲覧画面についても確認してみよう

FIAPSimpleSCADA設定ファイルの編集
備考：今回は設定ファイルはHTTPでも見れる

例：グループ1の場合は http://133.11.168.3/group01/fiapSimpleSCADA.xml

(ただし、通常は設定ファイルを見せたままの運用は行わない)



本日(16日)のスケジュール

10:00　IEEE1888レジストリの仕組み

10:30　レジストリを用いたCEMS実現への取組み

11:00　IEEE1888レジストリとの通信プログラミング実習

12:00　昼食

13:00　IEEE1888システムの設計と構築：その１

14:30　休憩

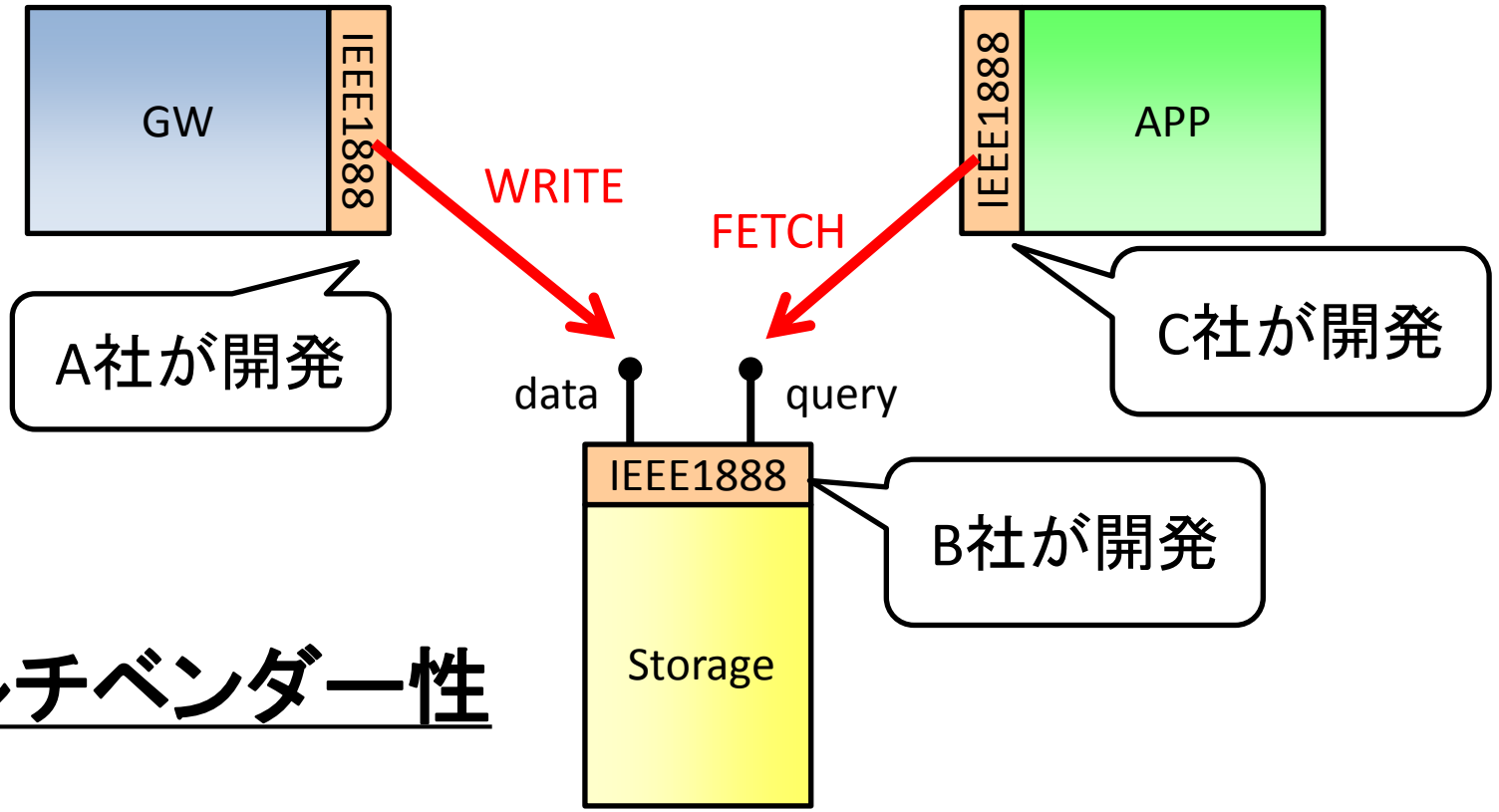
15:00　IEEE1888システムの設計と構築：その２

16:00　簡易な相互接続実験

17:00　解散

相互接続試験 (Interoperability Test) とは？

異なる組織が開発したIEEE1888通信スタック同士で、互いに通信できることを検証する試験



マルチベンダー性

簡易な相互接続試験

- 例えば、電力需給情報サービス
  - SOAP-EPR：http://www.futaba-kikaku.jp/services/jyukyu.php
  - WSDL：http://www.futaba-kikaku.jp/services/jyukyu.php?wsdl

ポイント表

ポイントID	電力会社	内容	更新頻度
http://futaba-kikaku.jp/epco/tepco/demand	東京電力	需要実績(万kw)	5分間隔
http://futaba-kikaku.jp/epco/tepco/balance	東京電力	需要バランス(%)	5分間隔
http://futaba-kikaku.jp/epco/kepco/demand	関西電力	需要実績(万kw)	3分間隔
http://futaba-kikaku.jp/epco/kepco/balance	関西電力	需要バランス(%)	3分間隔
http://futaba-kikaku.jp/epco/hepco/demand	北海道電力	需要実績(万kw)	1時間間隔
http://futaba-kikaku.jp/epco/hepco/balance	北海道電力	需要バランス(%)	1時間間隔
http://futaba-kikaku.jp/epco/tohoku-epco/demand	東北電力	需要実績(万kw)	5分間隔
http://futaba-kikaku.jp/epco/tohoku-epco/balance	東北電力	需要バランス(%)	5分間隔
http://futaba-kikaku.jp/epco/kyuken/demand	九州電力	需要実績(万kw)	5分間隔
http://futaba-kikaku.jp/epco/kyuden/balance	九州電力	需要バランス(%)	5分間隔

(注)各ポイントは最新値のみを提供します(FETCH手順のみ)。

このサービスと連携できることを確認してみよう。

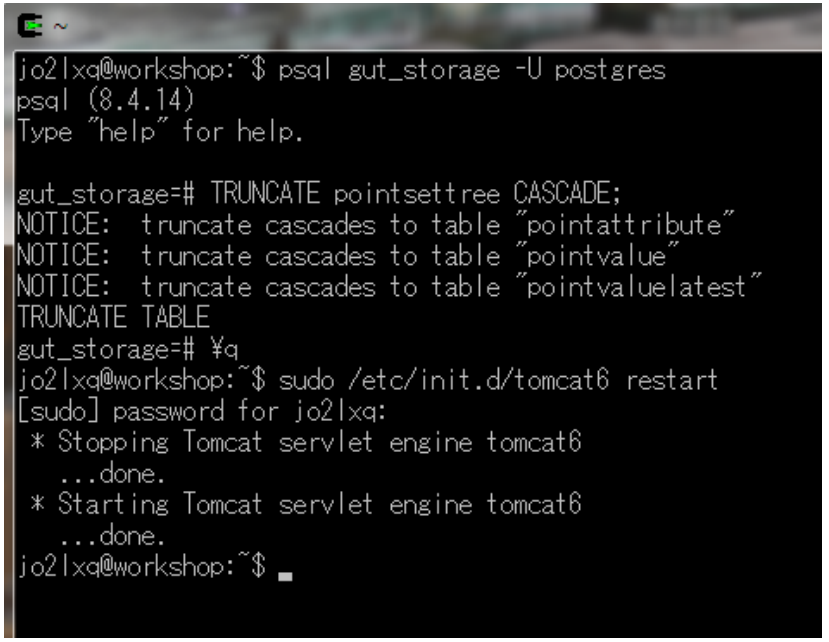
付録

FIAPStorageの初期化方法

実践ワークショップでは FIAPStorageの初期化方法に関する質問がありました。
以下が、その方法です（パスワードは、SDKを想定しています）。

ターミナルを開き、
shell $ psql gut_storage -U postgres
を実行すると、PostgreSQLサーバに接続される。その後、
gut_storage #　TRUNCATE pointsettree CASCADE;
によって、すべてのテーブルをクリアする。
gut_storage# ¥q
で、PostgreSQLサーバを抜け出す。

その後、tomcat6の再起動を行う。
$ sudo /etc/init.d/tomcat6 restart
[sudo password for gutp]:　　← gutpと入力
これでTomcat6が再起動され、初期化が完了する。



実行例: workshopで用いたサーバの場合