

# A New Architectural Approach for P1451.99 Binding to P1451.0: a First Proposal for P1451.99.0

R. Brama\*, P. Waher\*\*



# Motivation

- P1451.99 is a technology agnostic, harmonized *language* allowing a new degree of interoperability in the IoT
- P1451.99 has to be third-party also with respect to 1451.0
  - Easing adoption in existing deployed networks
  - Staging development effort
- Full 1451 integration becomes more difficult

A way to solve this problem, enabling different IoT verticals to become fully 1451.0 compliant, shall be found



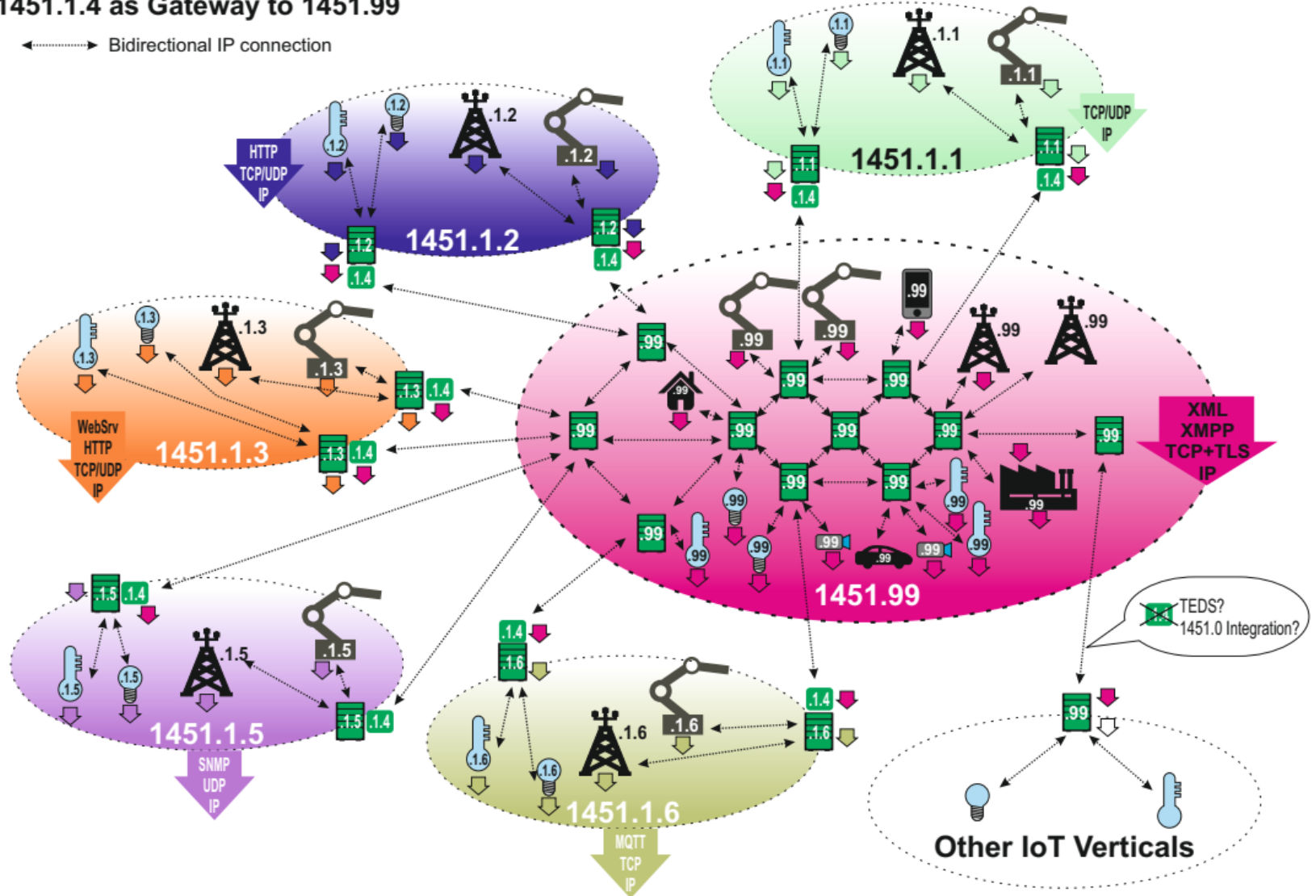
# Outline

- Current approach vs. New Proposal: unleashing 1451.1.4
- Specific Technology Abstraction
- One Standard, five possible implementations
- Today's Demonstration
- Demo Simulation Setup
- Conclusions

# Current Approach

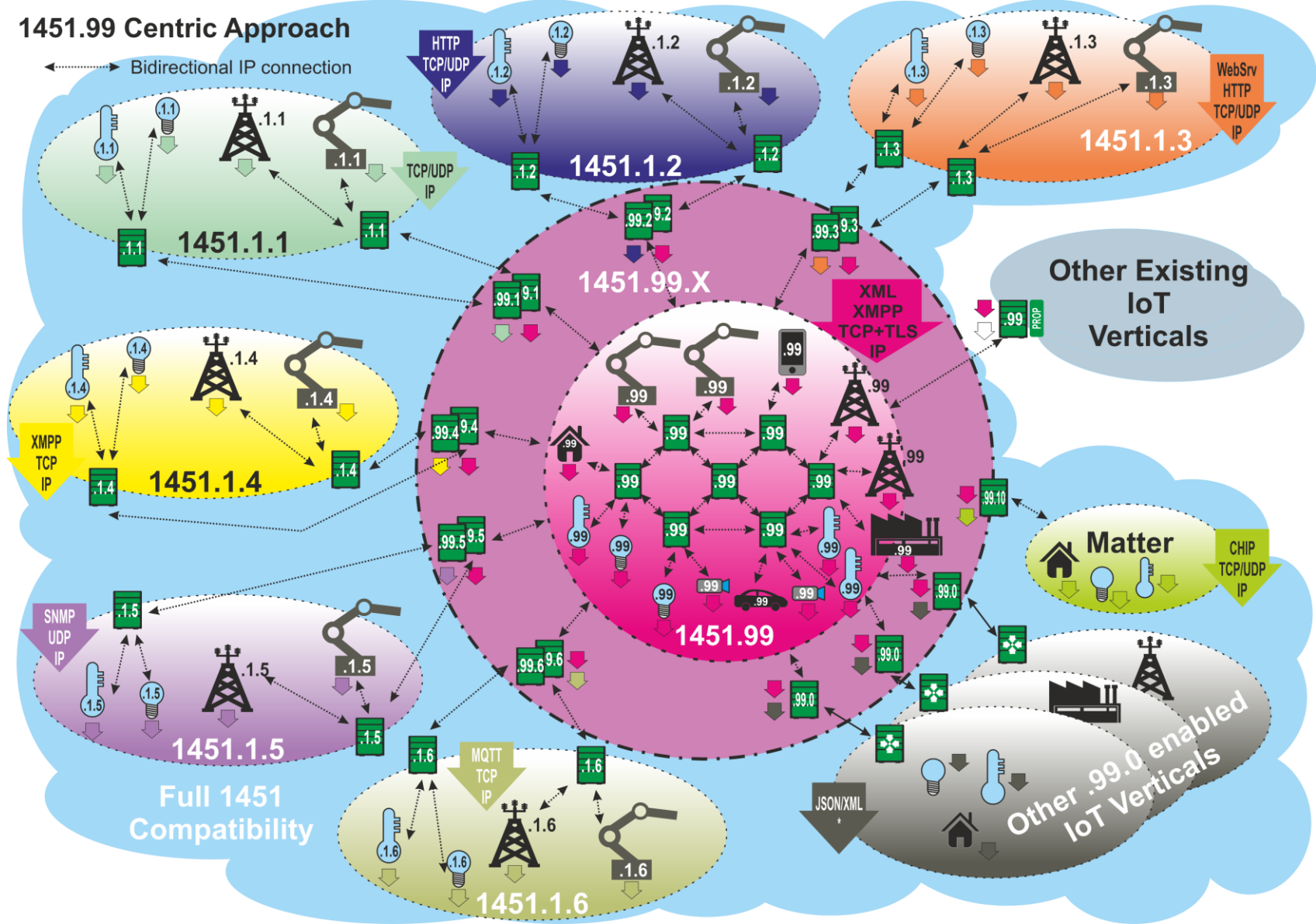
## 1451.1.4 as Gateway to 1451.99

↔ Bidirectional IP connection

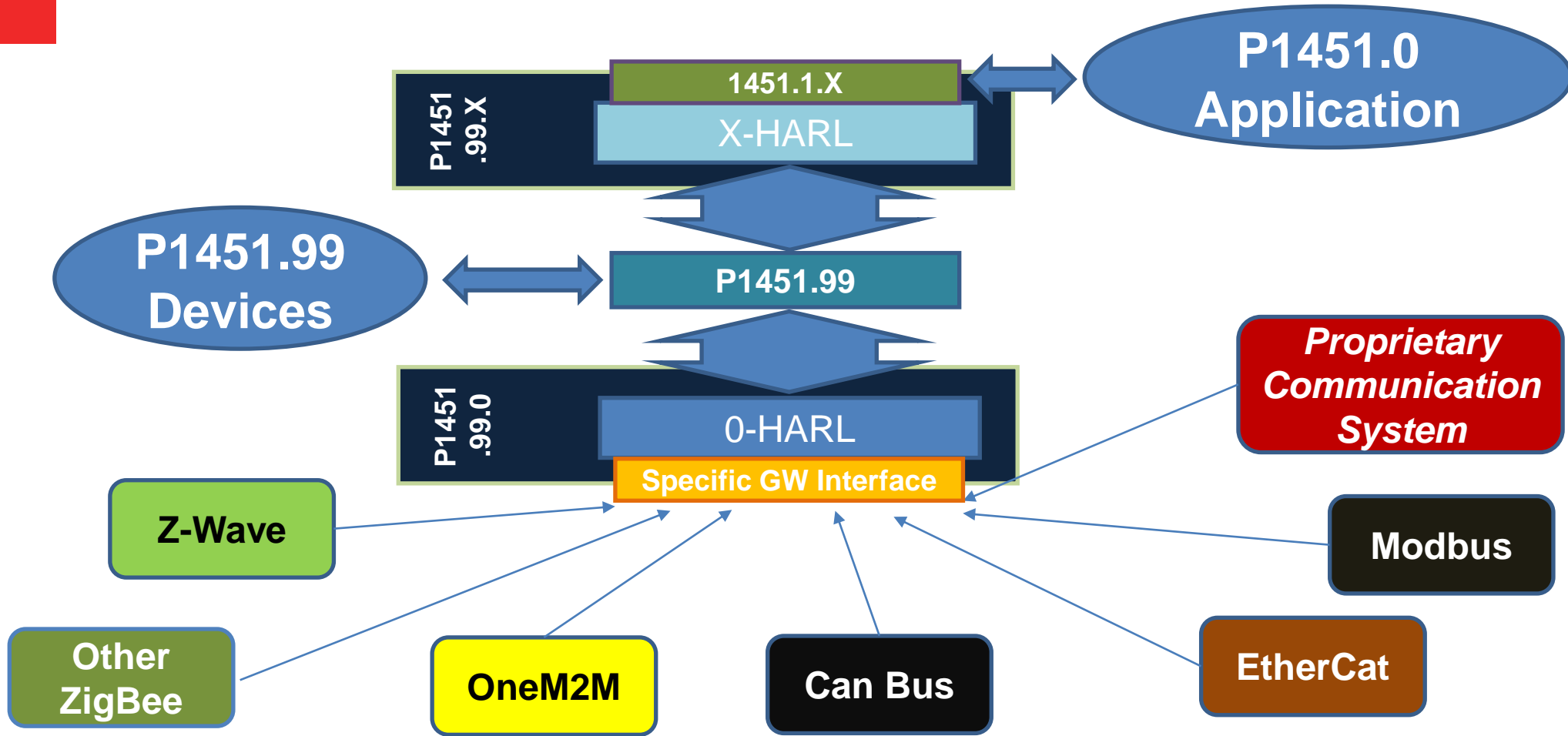


# Proposed Approach

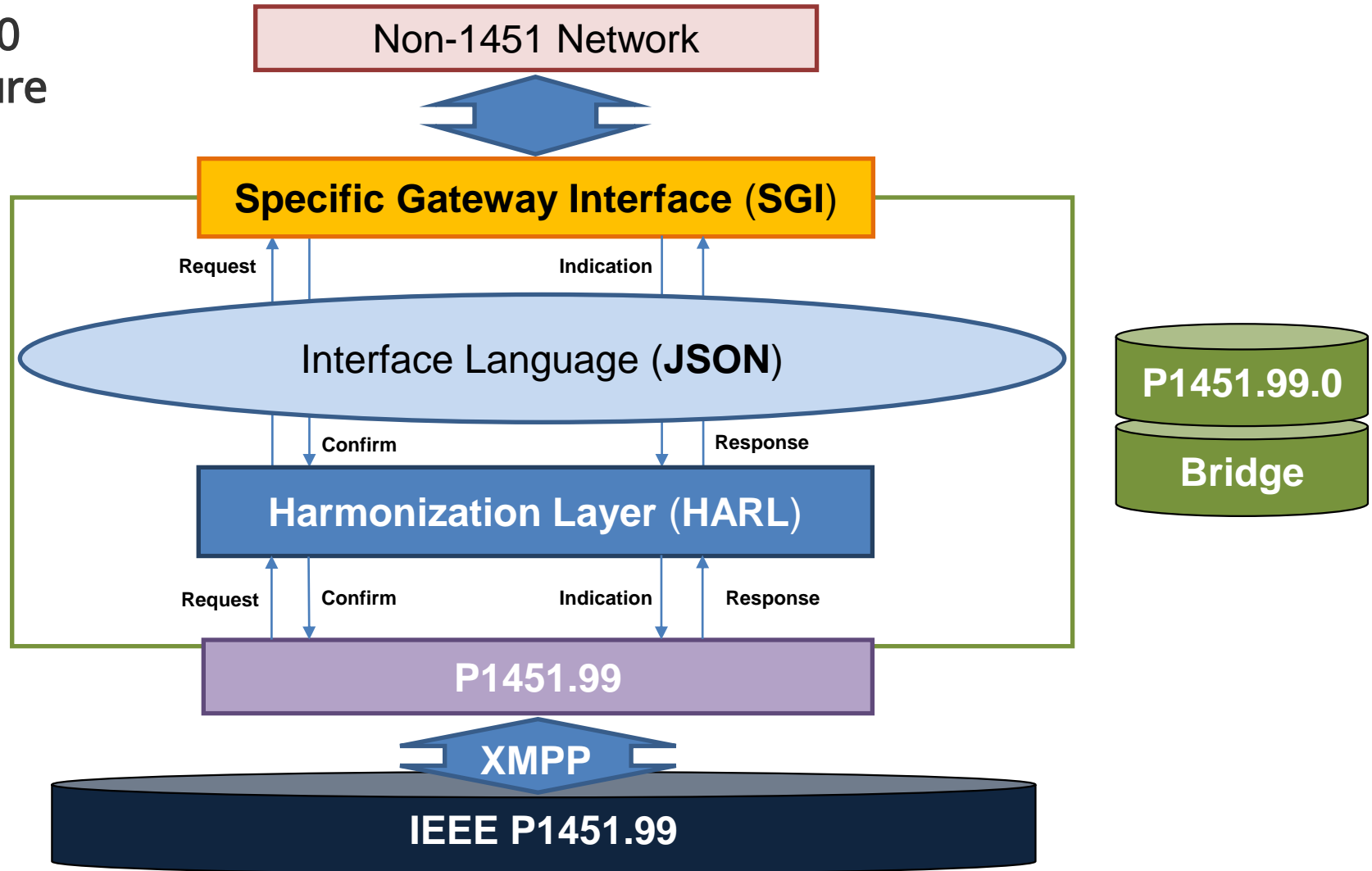
## 1451.99 Centric Approach



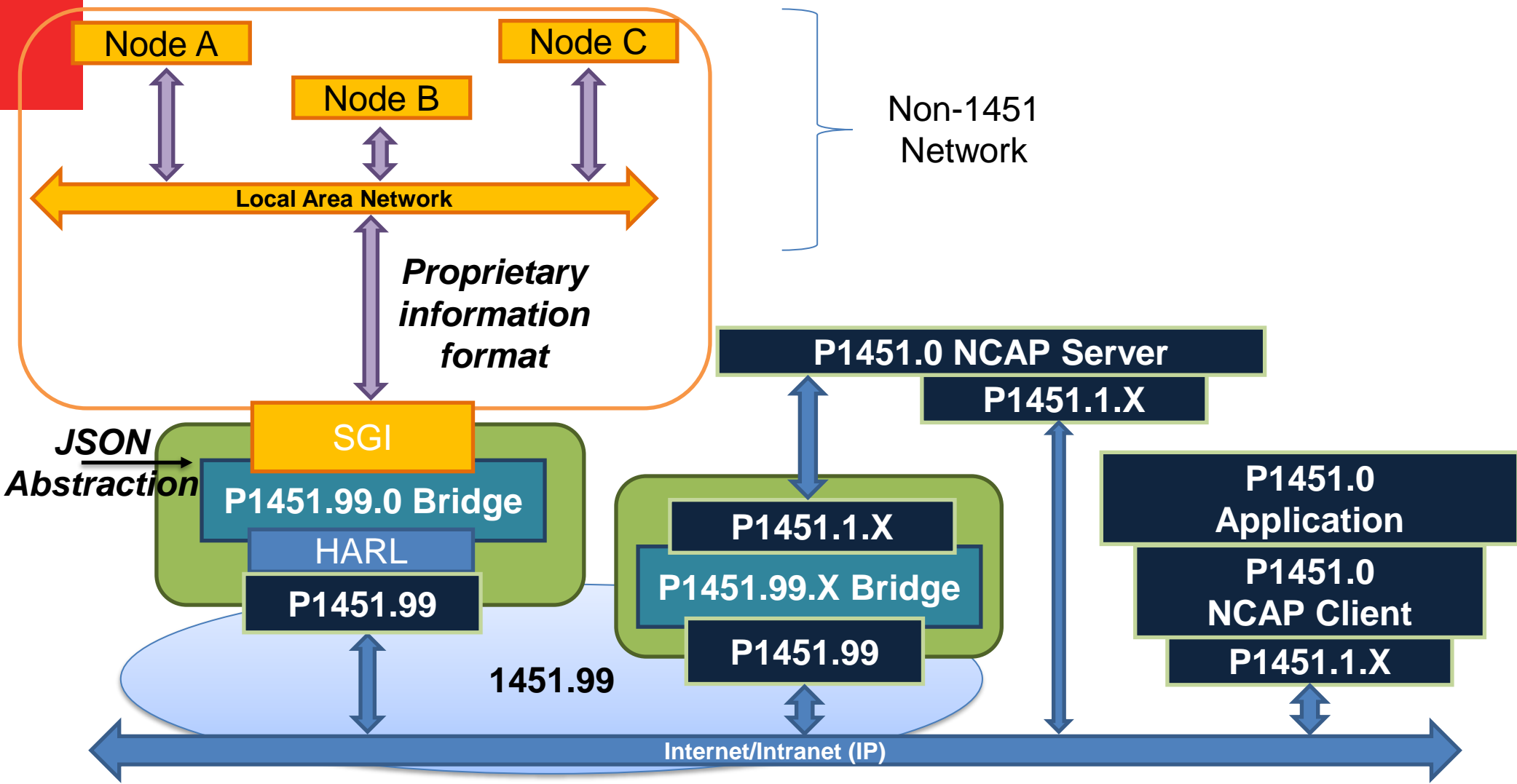
# Specific Technology Abstraction



# The Overall P1451.99.0 Architecture

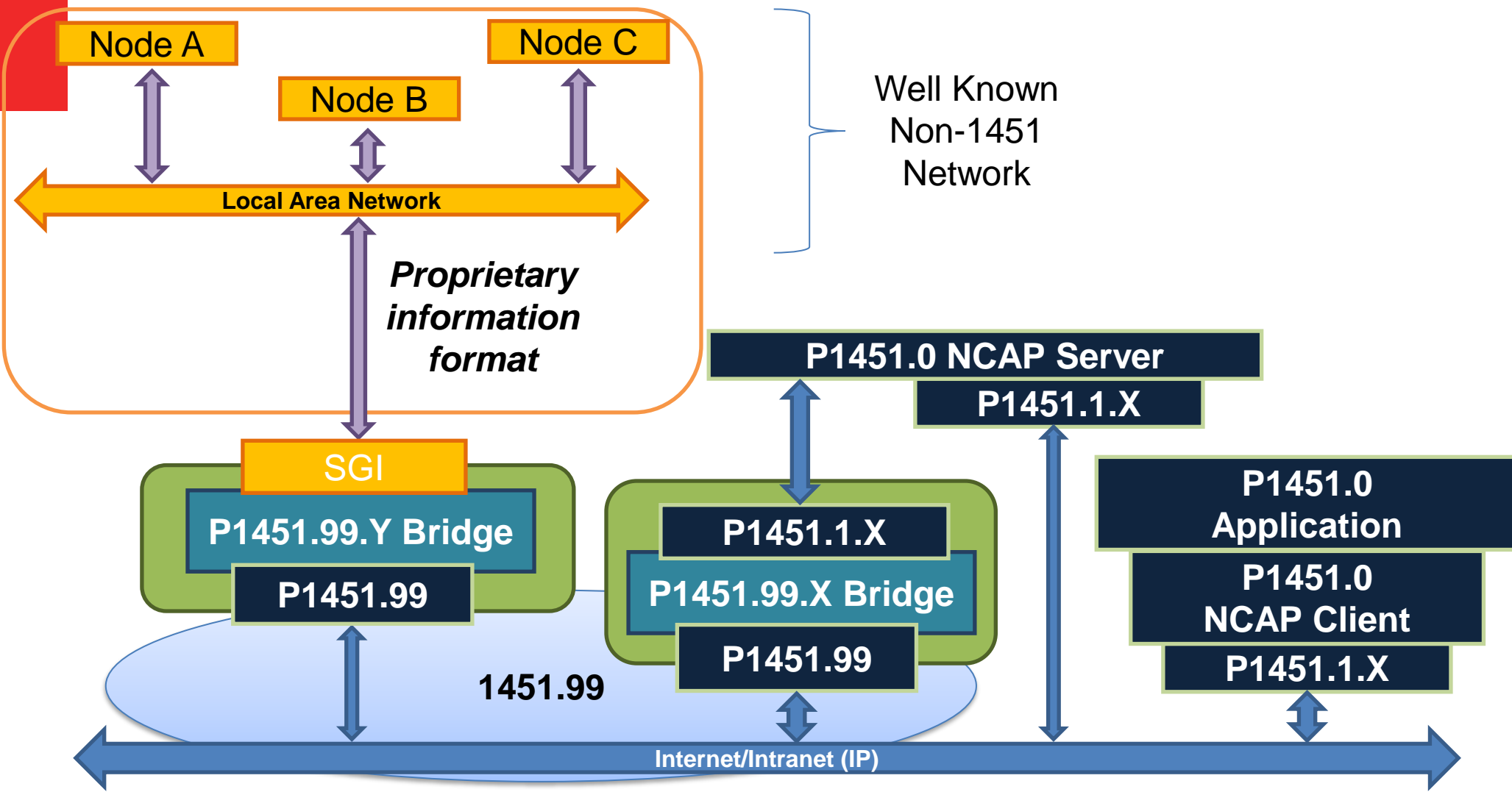


# Overall Communication Flow: arbitrary IoT Vertical case



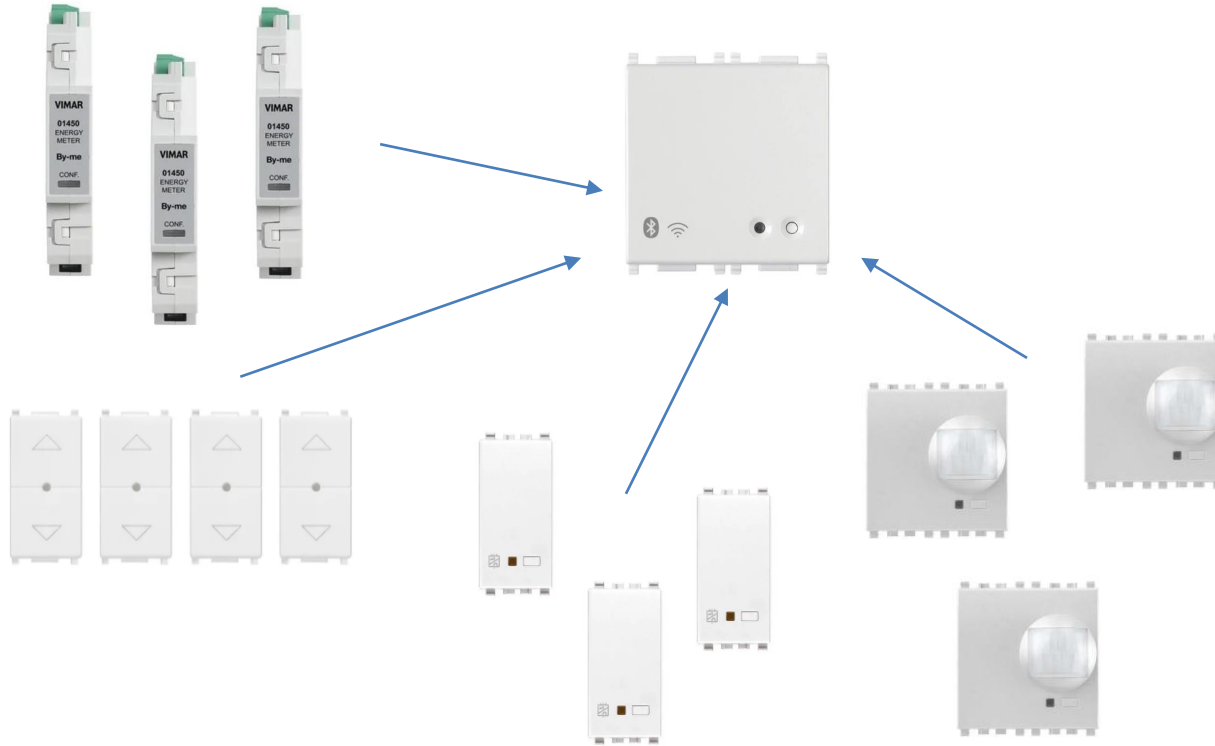


# Overall Communication Flow: well-known non-1451 IoT vertical



# One Standard, five possible implementations:

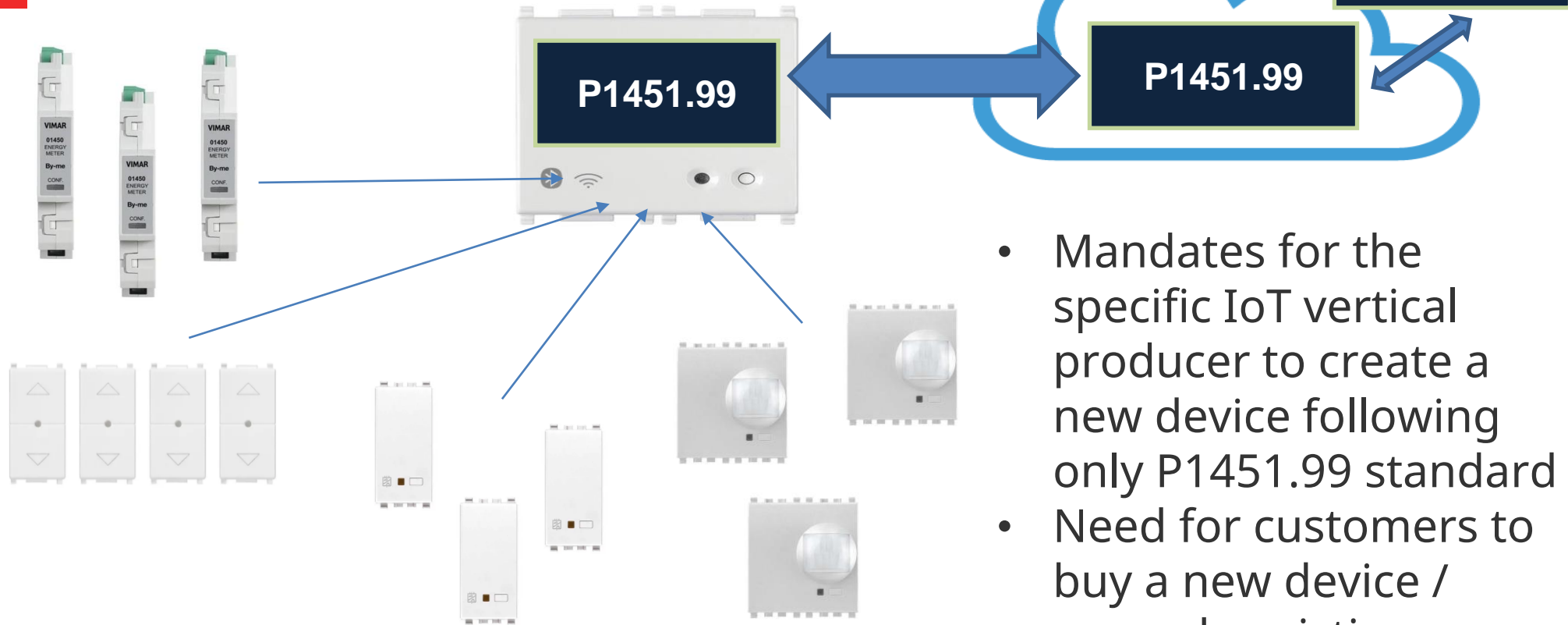
## 1. Partial Compliancy P1451.99 Bridge



- Existing IoT vertical is equipped with a gateway
- Gateway is substituted by a fully functional P1451.99 Bridge
- Only partial P1451.0 compliancy is obtained

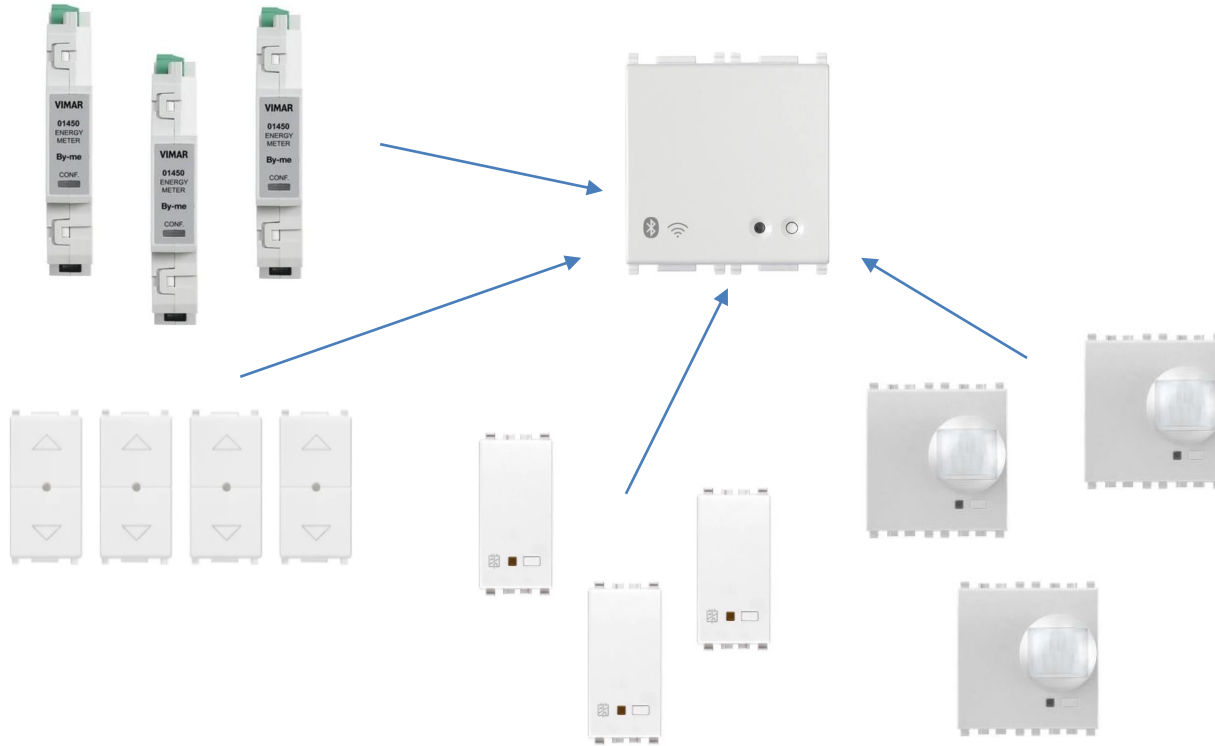
# One Standard, five possible implementations:

## 1. Partial Compliancy P1451.99 Bridge



# One Standard, five possible implementations:

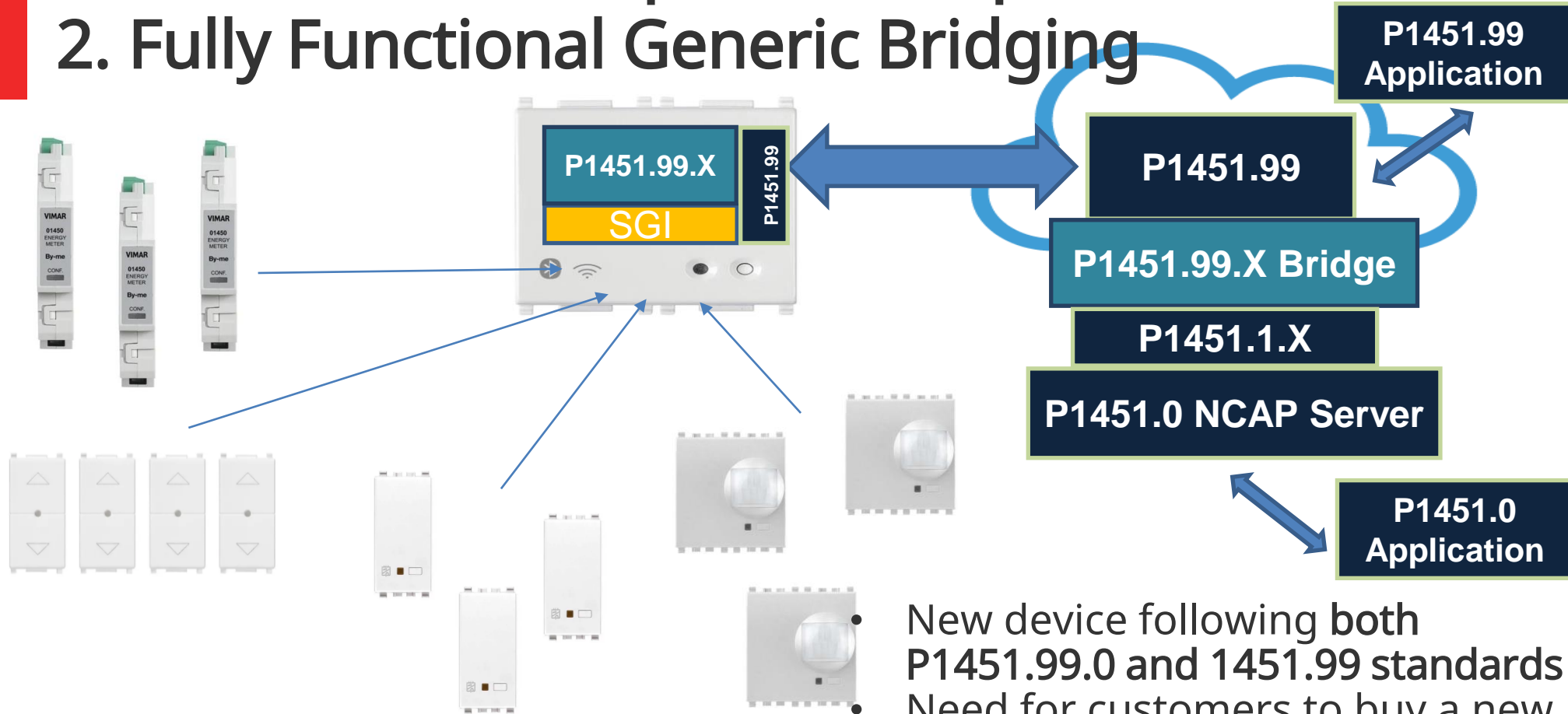
## 2. Fully Functional Generic Bridging



- Existing IoT vertical is equipped with a gateway
- Gateway is substituted with a generic P1451.99.0 + P1451.99 Bridge
- Complete P1451.0 compliancy is obtained

# One Standard, five possible implementations:

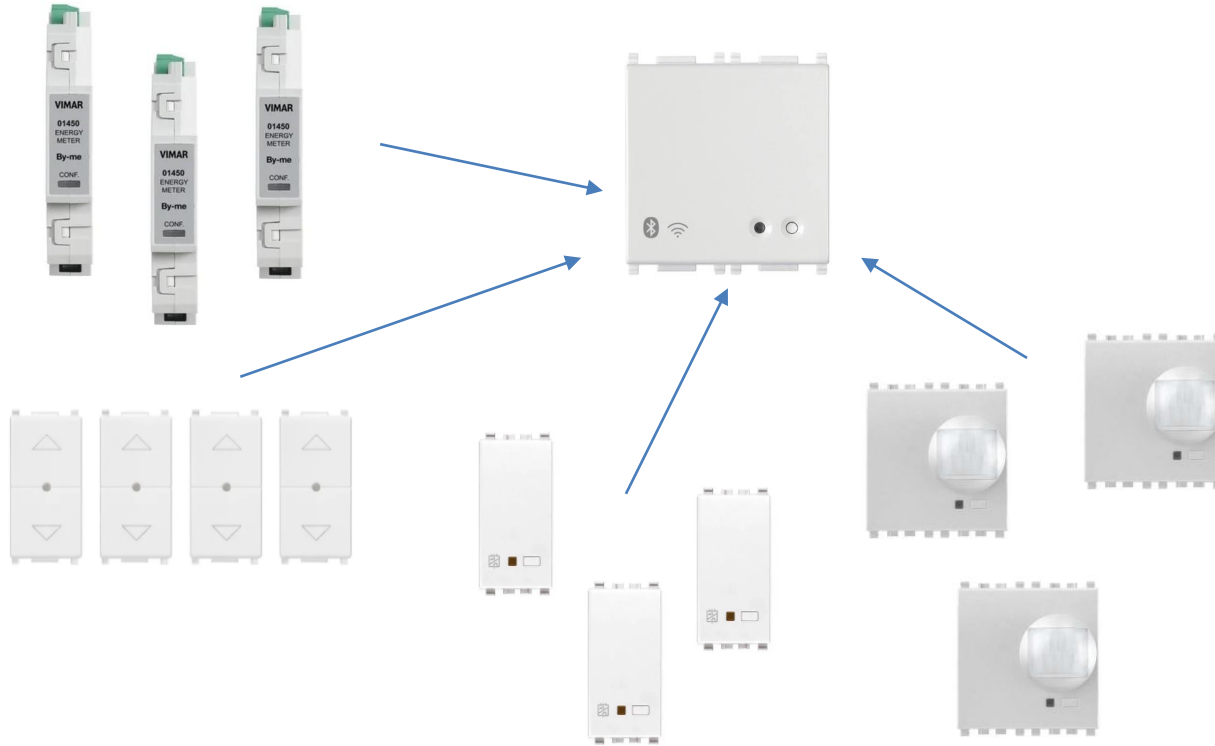
## 2. Fully Functional Generic Bridging



- New device following both P1451.99.0 and 1451.99 standards
- Need for customers to buy a new device / upgrade existing one

# One Standard, five possible implementations:

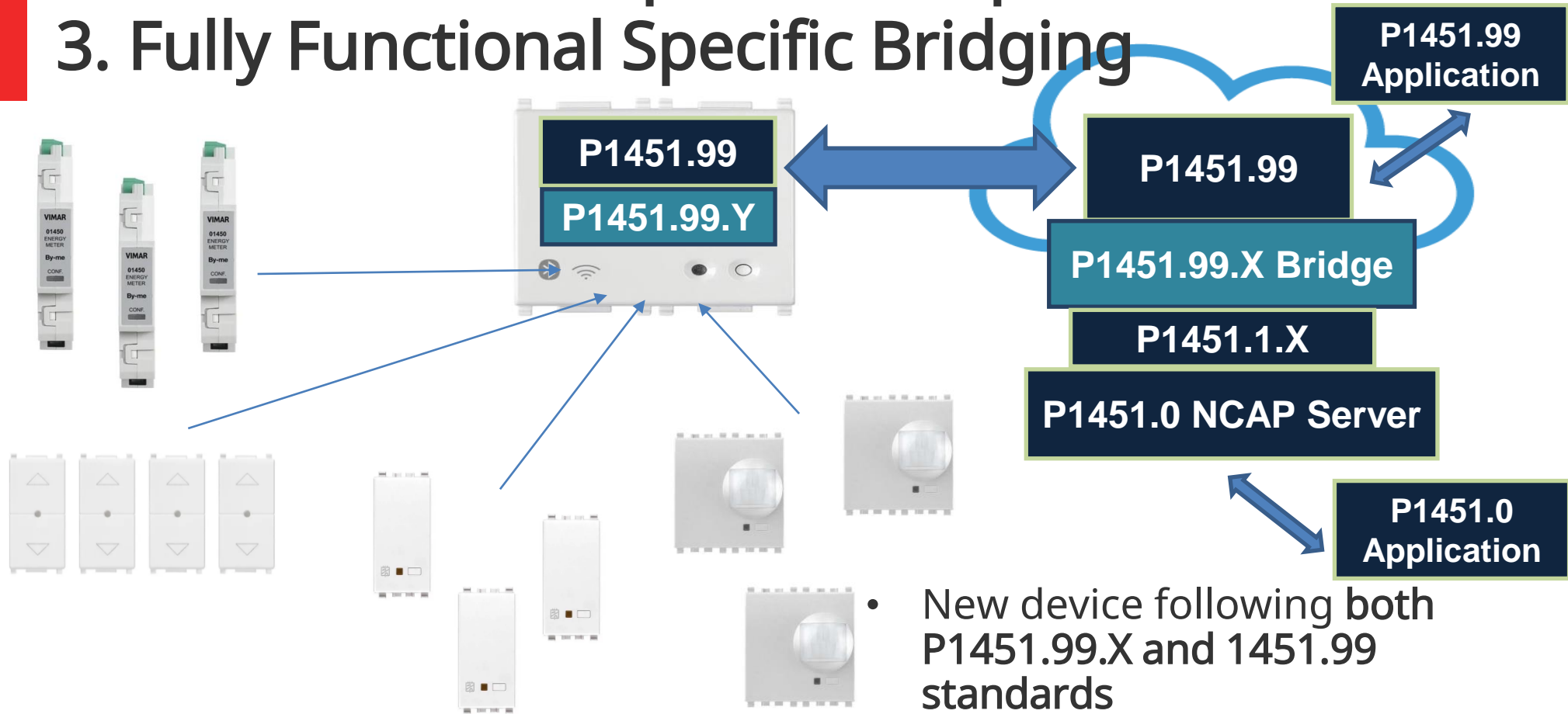
## 3. Fully Functional Specific Bridging



- Existing IoT vertical is equipped with a gateway
- Gateway is substituted with a specific P1451.99.X + P1451.99 Bridge
- Complete P1451.0 compliancy is obtained

# One Standard, five possible implementations:

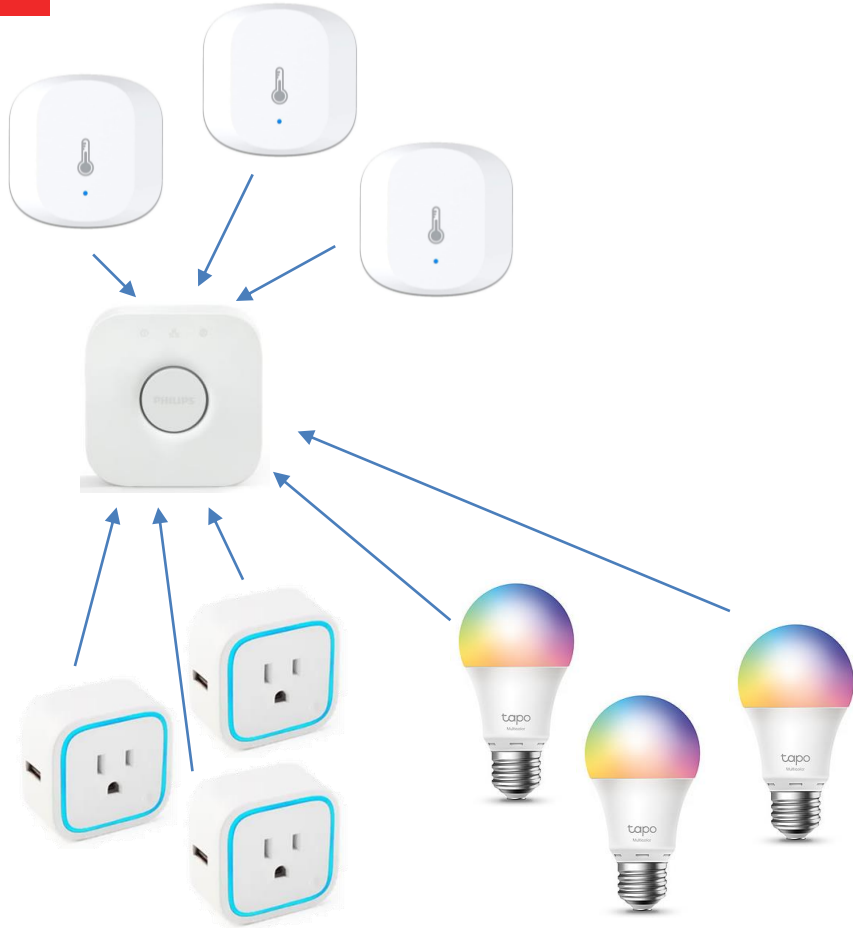
## 3. Fully Functional Specific Bridging



- New device following both P1451.99.X and 1451.99 standards
- Need for customers to buy a new device / upgrade existing one

# One Standard, five possible implementations:

## 4. Locally Assisted P1451.99 Gateway

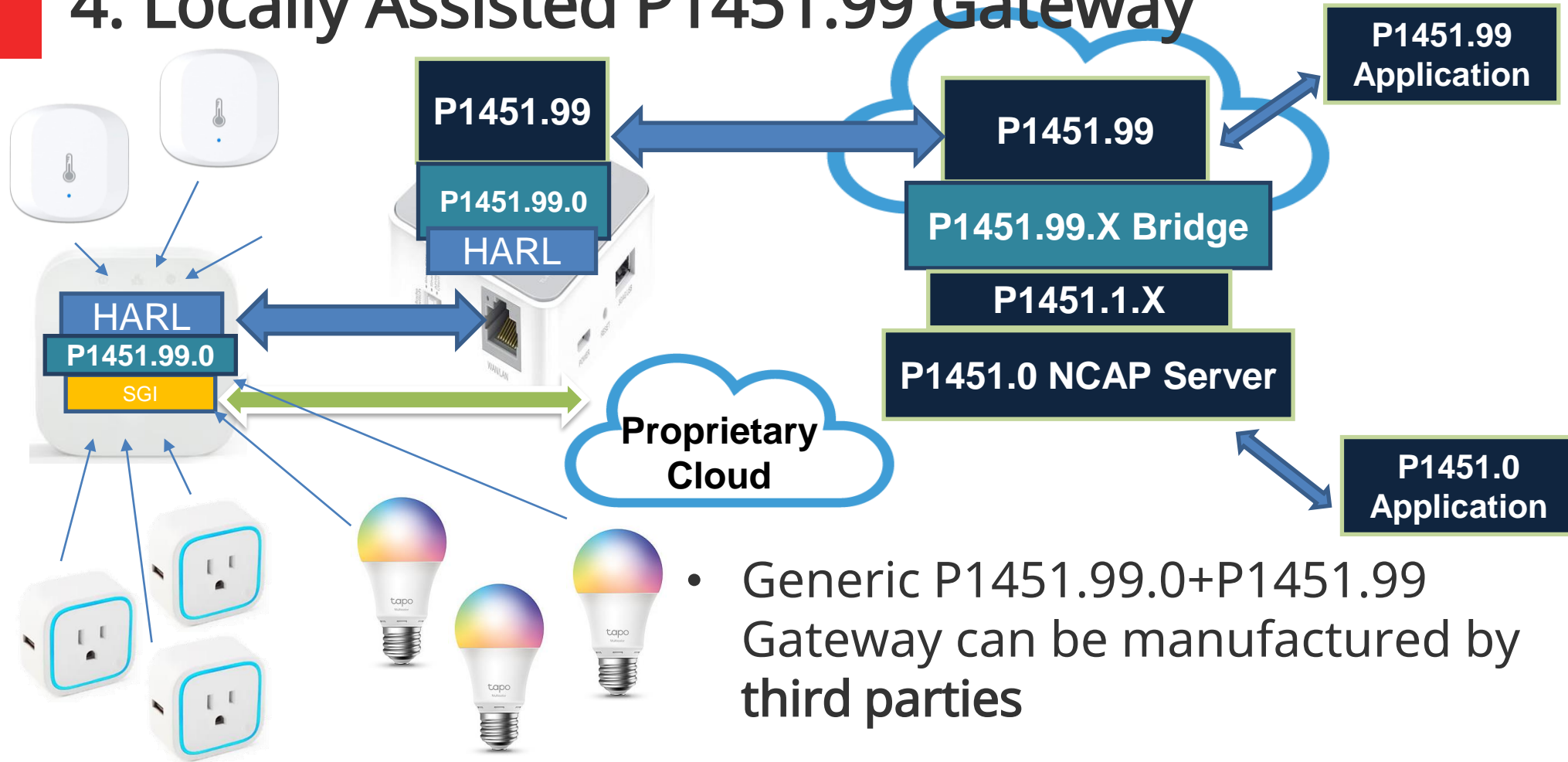


- Existing IoT vertical is equipped with a gateway
- The gateway is equipped with **P1451.99.0 only** by specific IoT vertical manufacturer
- Existing gateway is assisted by a **generic, LAN scope, gateway** implementing **P1451.99** on a network interface



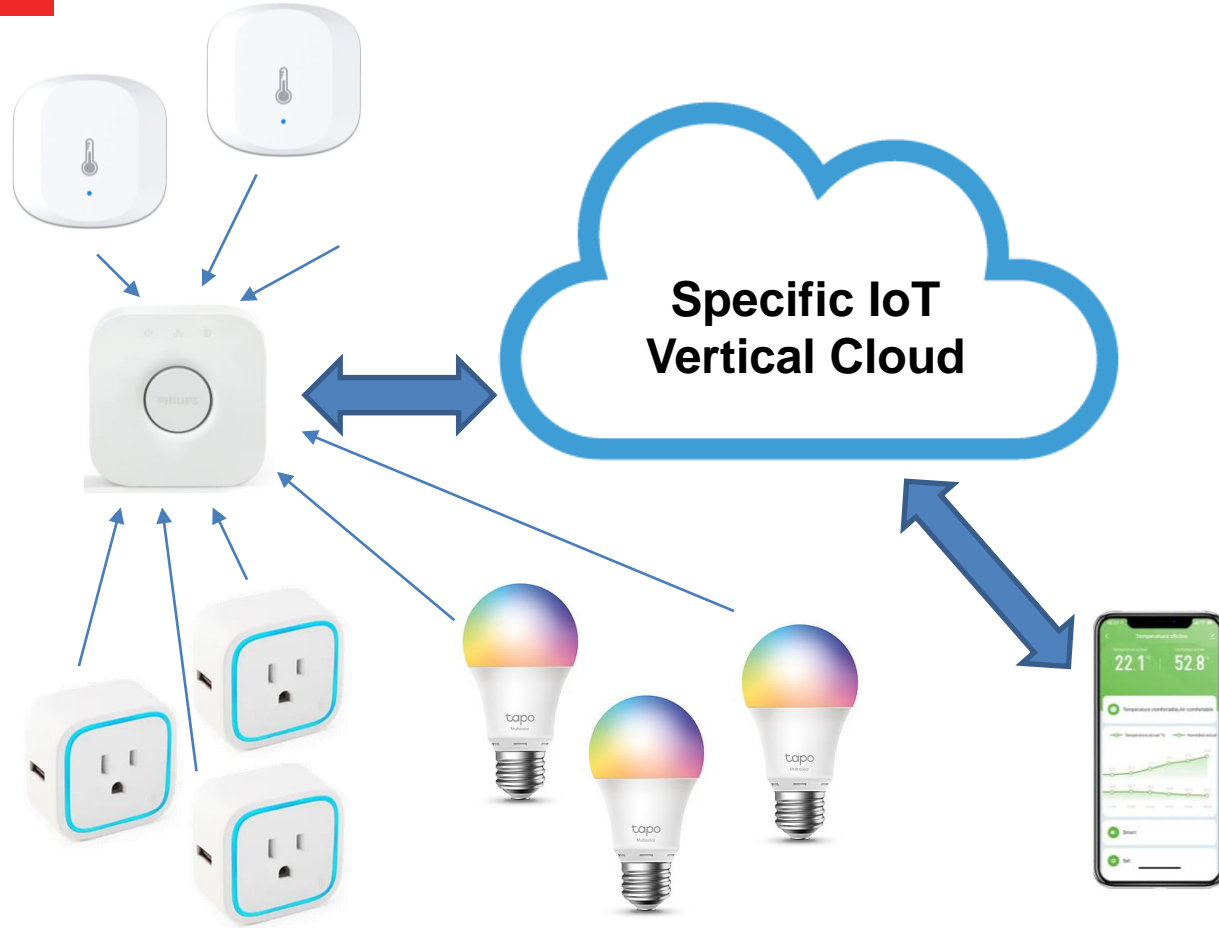
# One Standard, five possible implementations:

## 4. Locally Assisted P1451.99 Gateway



# One Standard, five possible implementations:

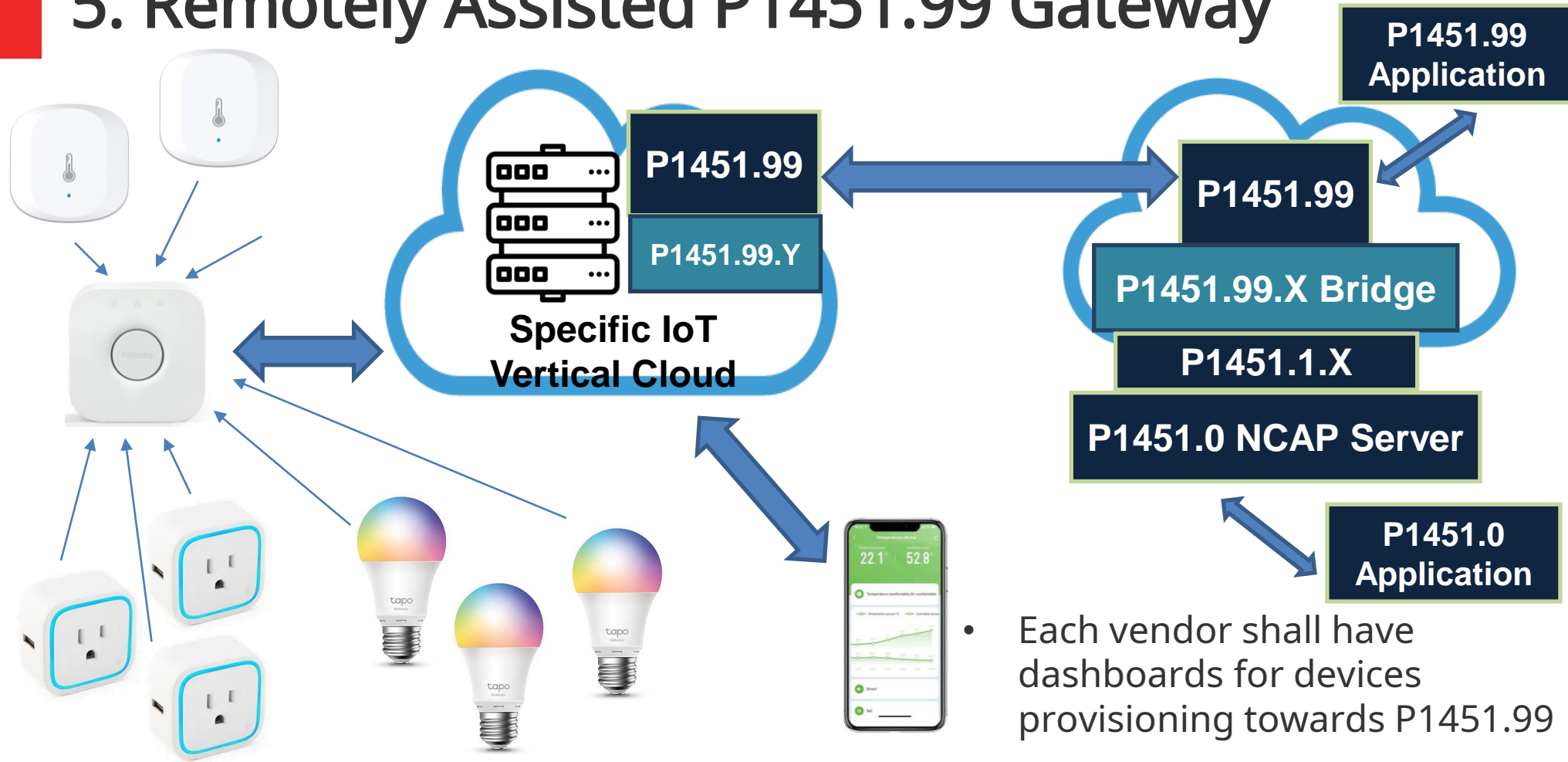
## 5. Remotely Assisted P1451.99 Gateway




- Existing IoT vertical is equipped with a gateway and a proprietary cloud service
- The full P1451.99.X + P1451.99 bridging is realized **on the cloud** as a remote service

# One Standard, five possible implementations:

## 5. Remotely Assisted P1451.99 Gateway





# One Standard, five possible implementations:

## 5. Remotely Assisted P1451.99 Gateway

- No need to update existing hardware and software: P1451.99 and P1451.0 full compliancy is provided as a service
- Multi-vendor approach
- Fastest adoption but lack of uniformity
- A P1451.0 compliant application can be used to interface with whatever P1451.99 harmonized IoT vertical

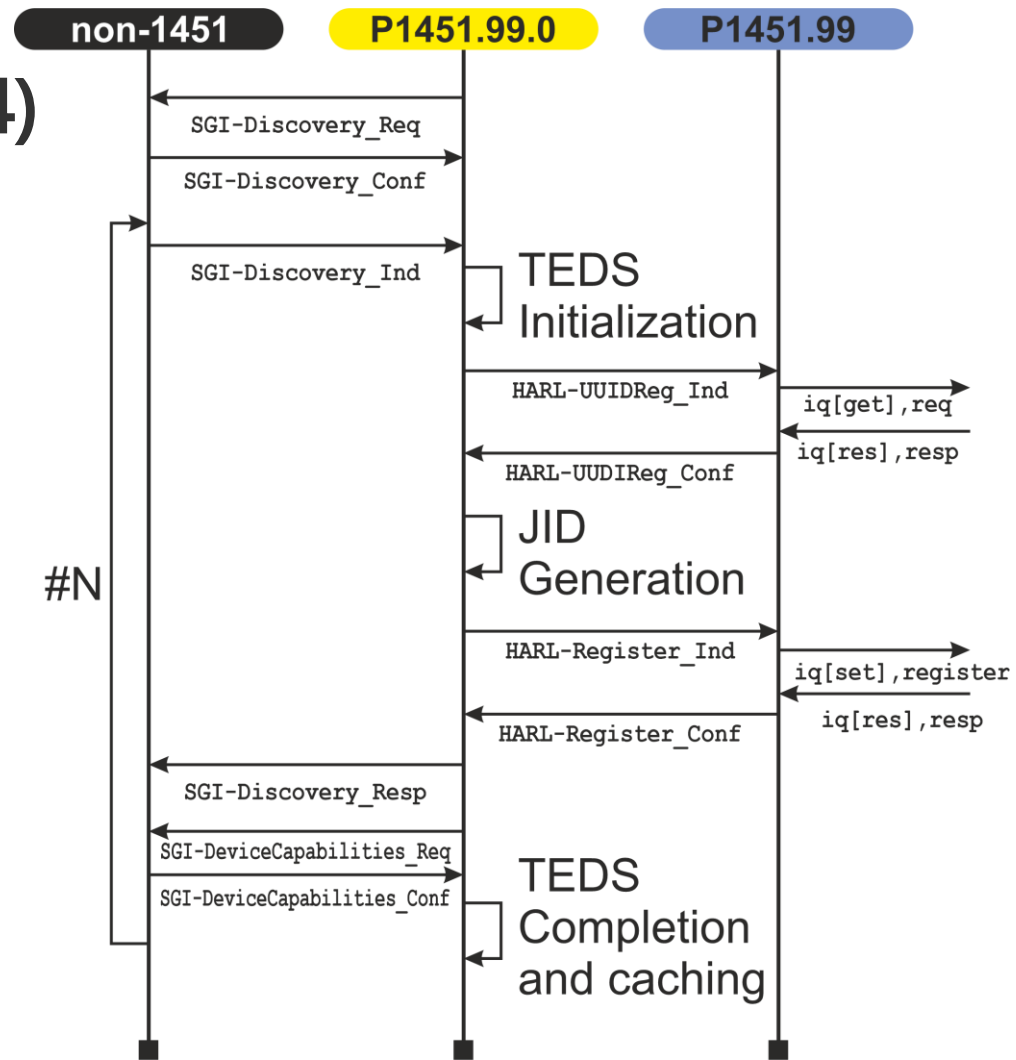


# Today's Demonstration (1/4)

- We aim to show how it is possible to obtain full P1451.0 compliancy by exploiting a P1451.99.0 gateway implementing SGI and HARL with JSON data representation
- Targets of the demonstration:
  - UUID, SGIID and JID matching;
  - Simulated readings from non-1451 devices towards P1451.99 ecosystem;
  - Simulated actuation of non-1451 devices by means of a P1451.99 actuation command;

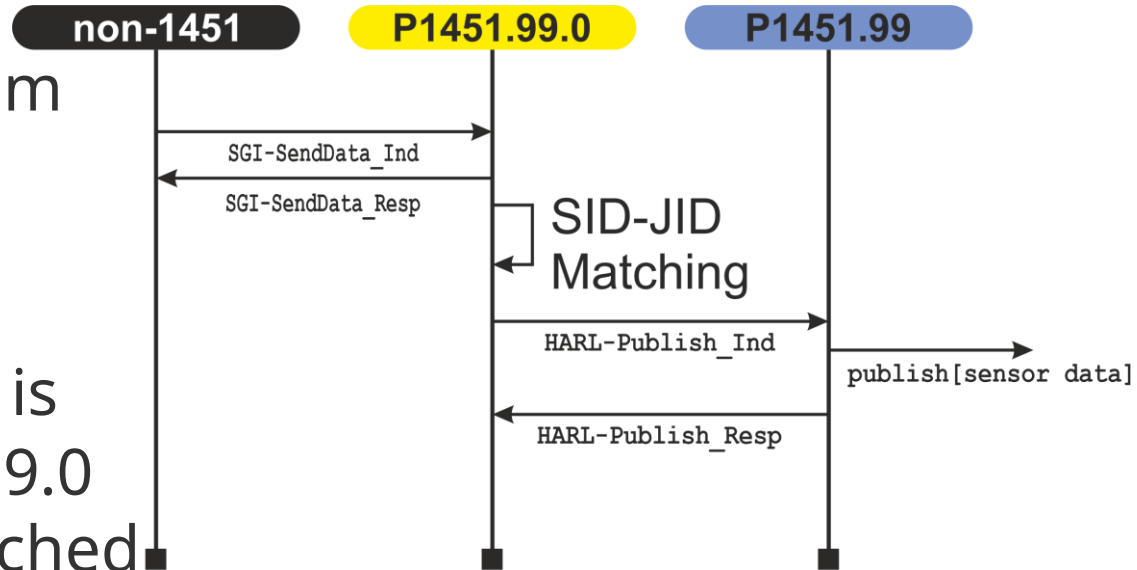
# Today's Demonstration (2/4)

- Simulated non-1451 devices will initiate discovery on request;
- Each SGI-Discovery\_Ind primitive generates a TEDS
- UUID within the TEDS is retrieved from P1451.99 network (P1451.99.X service)
- JID is matched with UUID at P1451.99.X level
- TEDS is completed



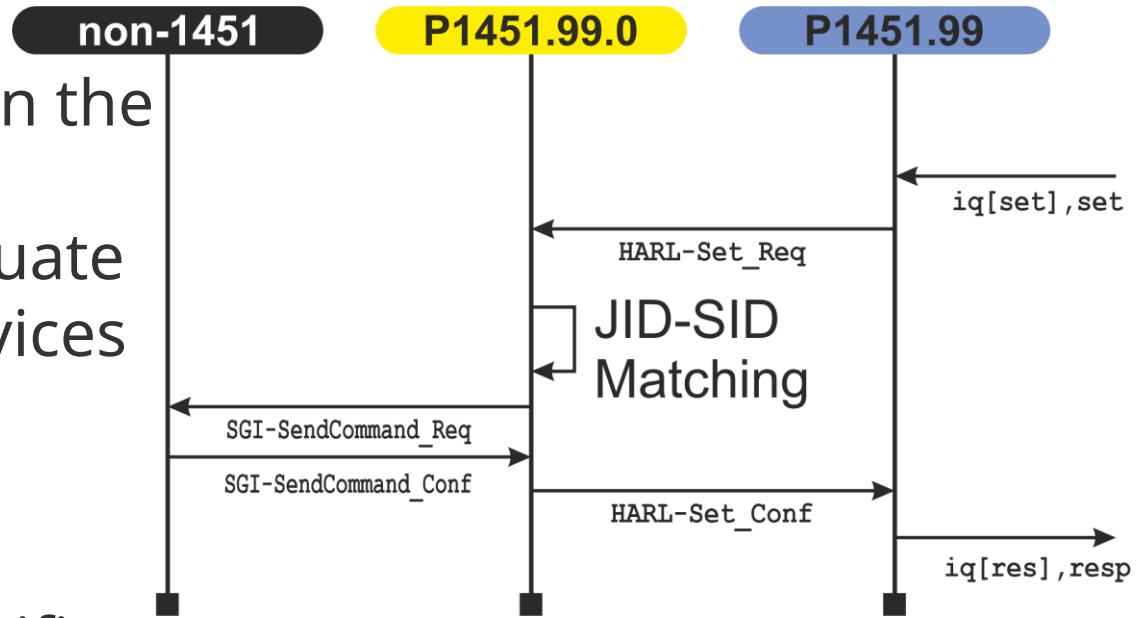
# Today's Demonstration (3/4)

- A reading is available from non-1451 network. It generates a SGI-SendData\_Ind
- An HARL-Publish\_Ind is triggered by the P1451.99.0 gateway after SID is matched with JID
- This generates a P1451.99 publication



# Today's Demonstration (4/4)

- A set request received on the P1451.99 harmonized network requests to actuate one of the non-1451 devices
- An `HARL-Set_Req` is triggered towards the P1451.99.0 gateway
- JID is matched with Specific ID (address used on non-1451 network) to generate a `SGI-SendCommand_Req`





# Demo Simulation Setup

- A web server with REST interface will be offering *primitives* allowing to exchange JSON objects
- Three primitives are available:
  - SGI-Discovery-Ind
  - SGI-SendData\_Ind
  - HARL-Set\_Req
- When user posts on those methods a JSON object the simulated P1451.99.0 gateway replies with the appropriate action



# Conclusions

- A new approach to P1451.99 architecture, based on P1451.99.X bindings, has been proposed
- It has been demonstrated that the most abstract binding allows interfacing an arbitrary non-1451 vertical with P1451.99
- We're looking for interested players to start P1451.99.0 and to implement the first, JSON based, fully functional generic bridge