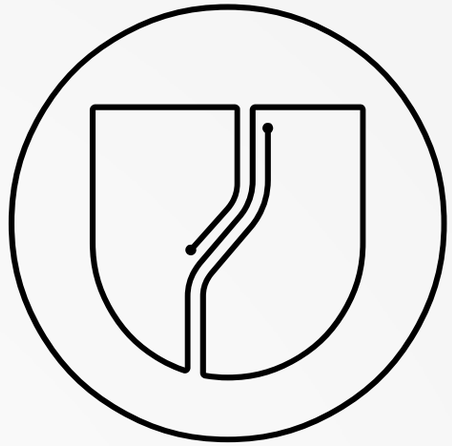
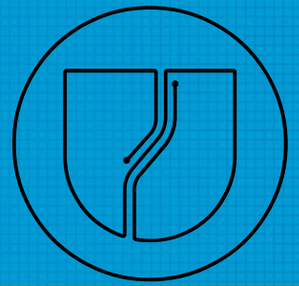


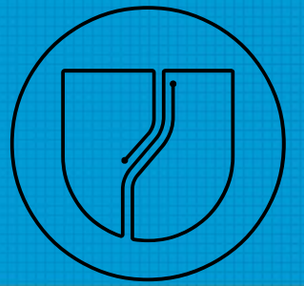
Evolution of (OTA) Update in the IoT world

About me

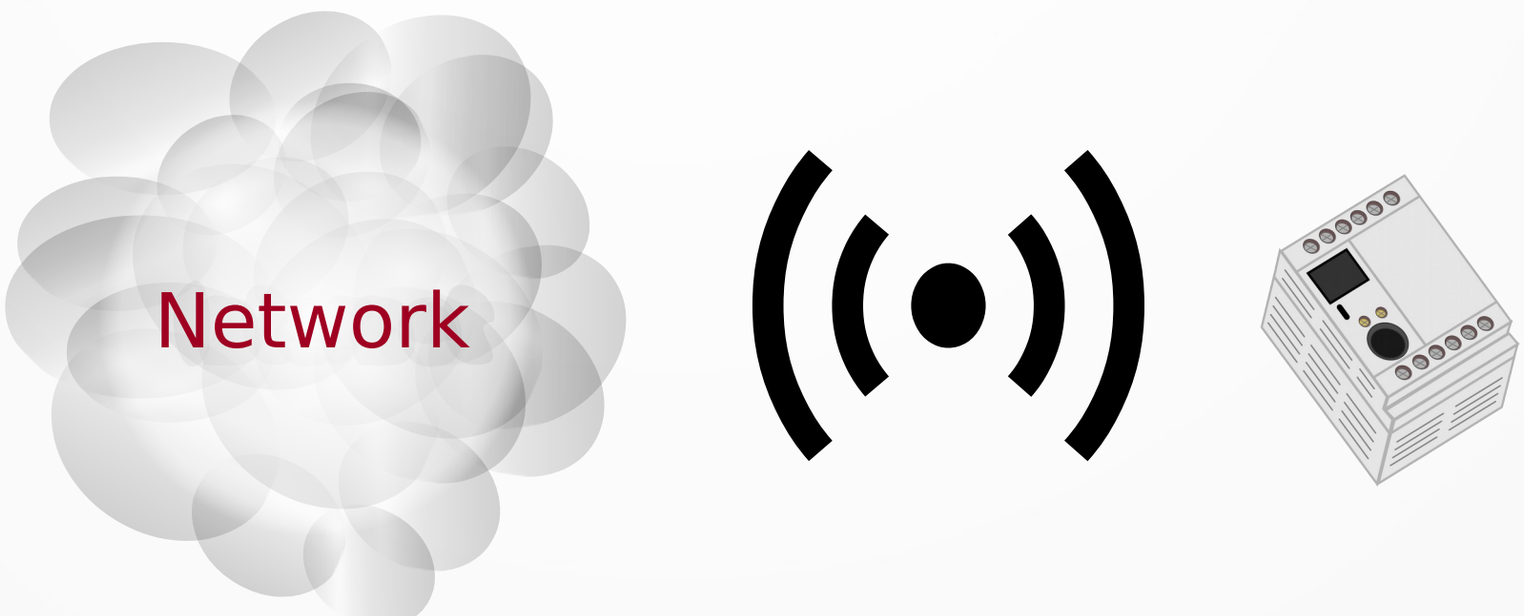


- Freelancer Embedded Developer
- U-Boot Custodian for NXP‘s i.MX
- Author and Maintainer of SWUpdate
- Focus on Linux Embedded

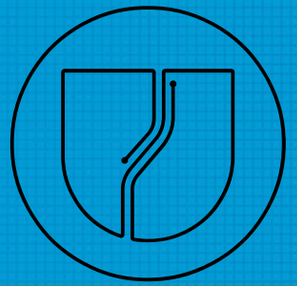
OTA : Over The Air



Who we should thank for „Over the Air“ ?



OTA Update



- It must be „OTA“
  - Over the Air is the keyword !
- But really it is not provided by an update agent
- It is provided by Chip Manufacturer (Atheros, ..)
- It is provided by drivers in kernel
- It is provided by infrastructure (WPA2, etc.)

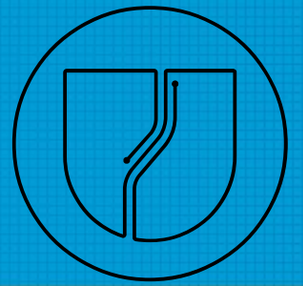
An „OTA“ updater just uses sockets...

UTW Update

UTW: Update “Under the Water” !!! (TM)

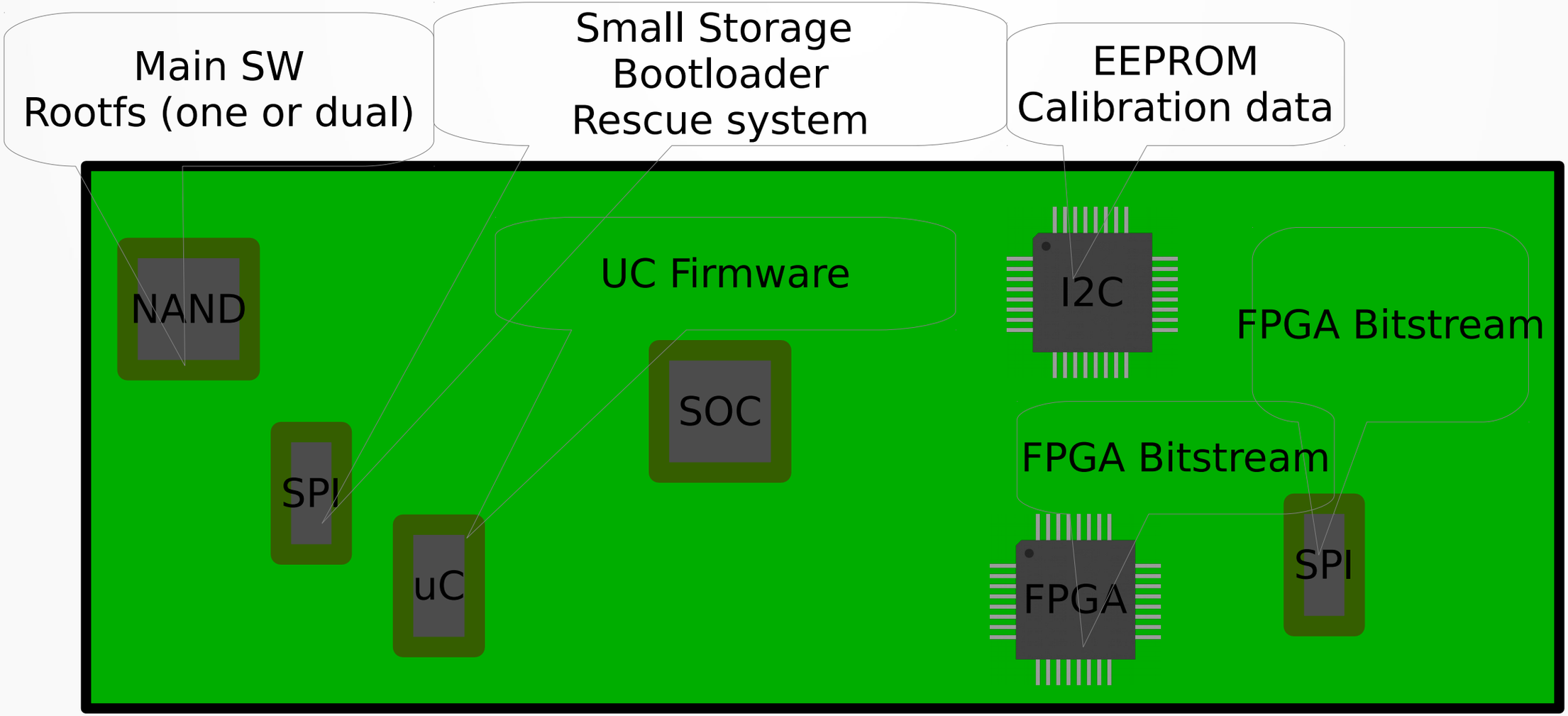
https://www.blueyerobotics.com/

Features for an update agent



- Limited bandwith
  - Resume a broken connection
  - Delta updates (rdiff handler in SWUpdate)
  - Compressed images
- Security :
  - Signed images (Keys, certificates)
  - Encrypted Images
  - Set of algorithms
  - No downgrading
  - Audit by external security companies

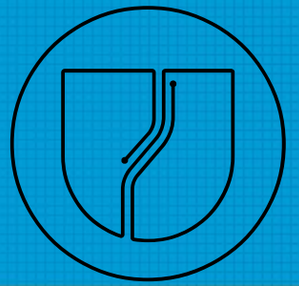
Update all components



SWUpdate chosen by CIP

CIVIL
INFRASTRUCTURE
PLATFORM

Bad points - wishes



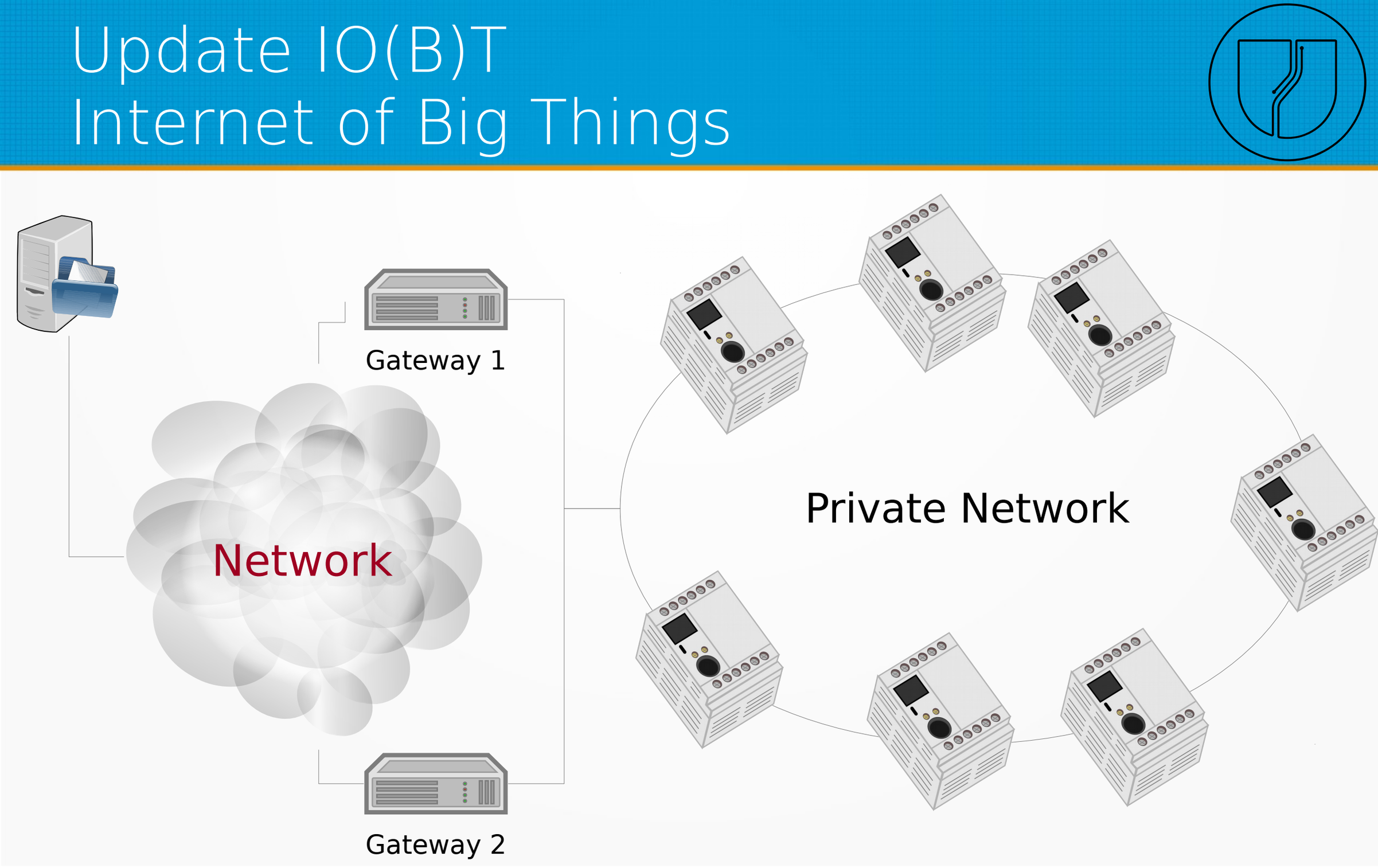
- Hard to manage updates from v1 to v5 directly
- Depends on a u-boot library that needs to be rebuilt for each target
  - (replacement https://github.com/sbabic/libubootenv).
  - SWUpdate 2019.04 with CONFIG_UBOOT_NEWAPI will make use of the standalone libubootenv library, and will read default initial environment from “/etc/u-boot-initial-env”.

https://wiki.linuxfoundation.org/civilinfrastructureplatform/cip_comparison_report

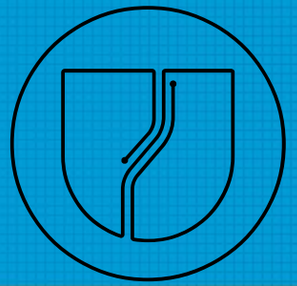
CIVIL INFRASTRUCTURE PLATFORM

System Update

Update IO(B)T
Internet of Big Things

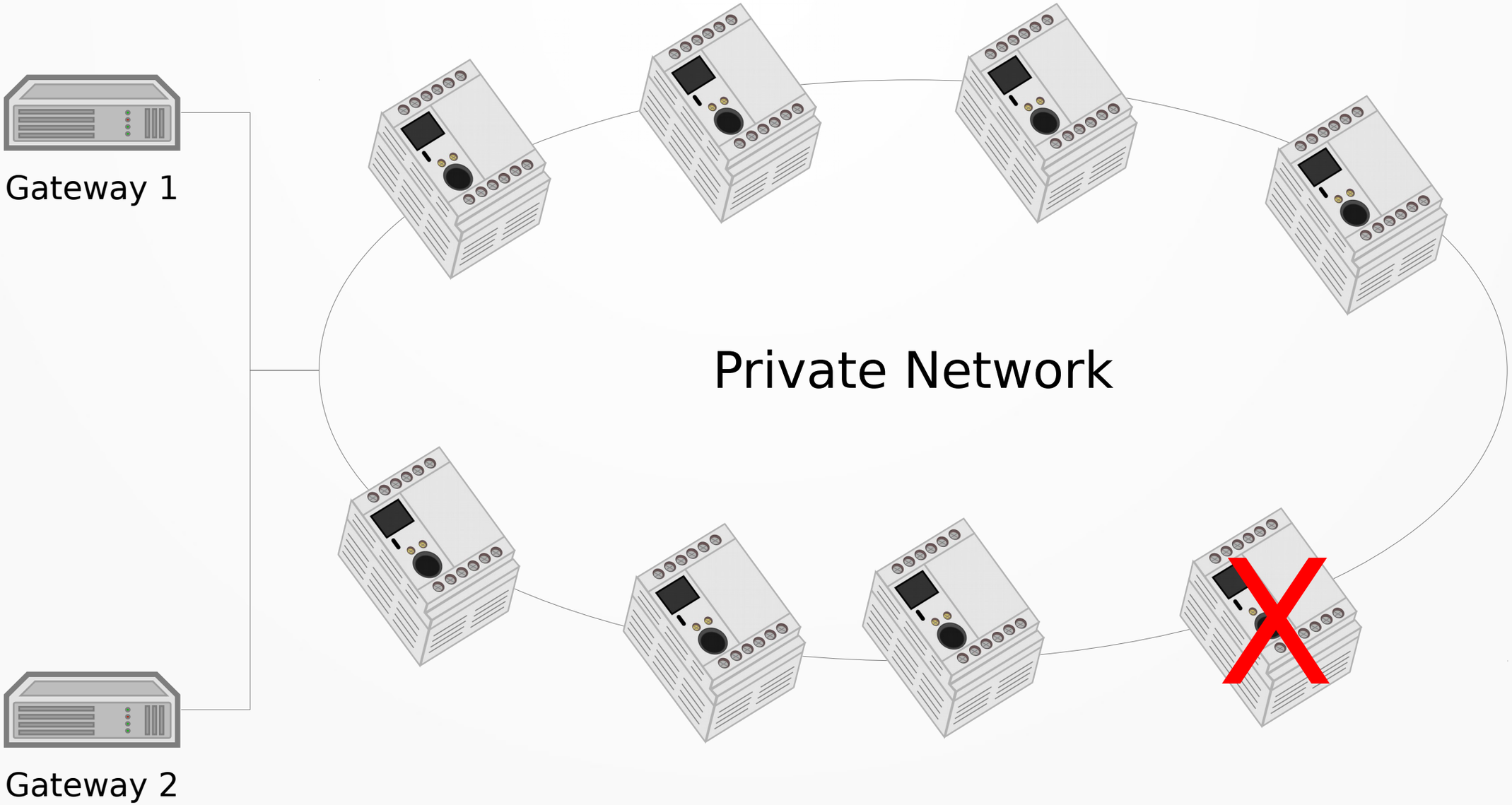


Requirements

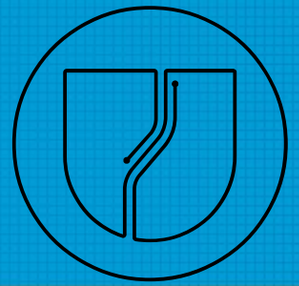


- Same image for all devices
  - Do not duplicate SWUs
- Streaming of Software
- Update devices in parallel
- Detect topology by Update
- Check successful update and initiate a network restart (network update is „atomic“)
- Still allow single update in private network

Automatic SW align

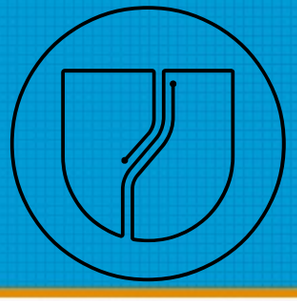


More services for Update



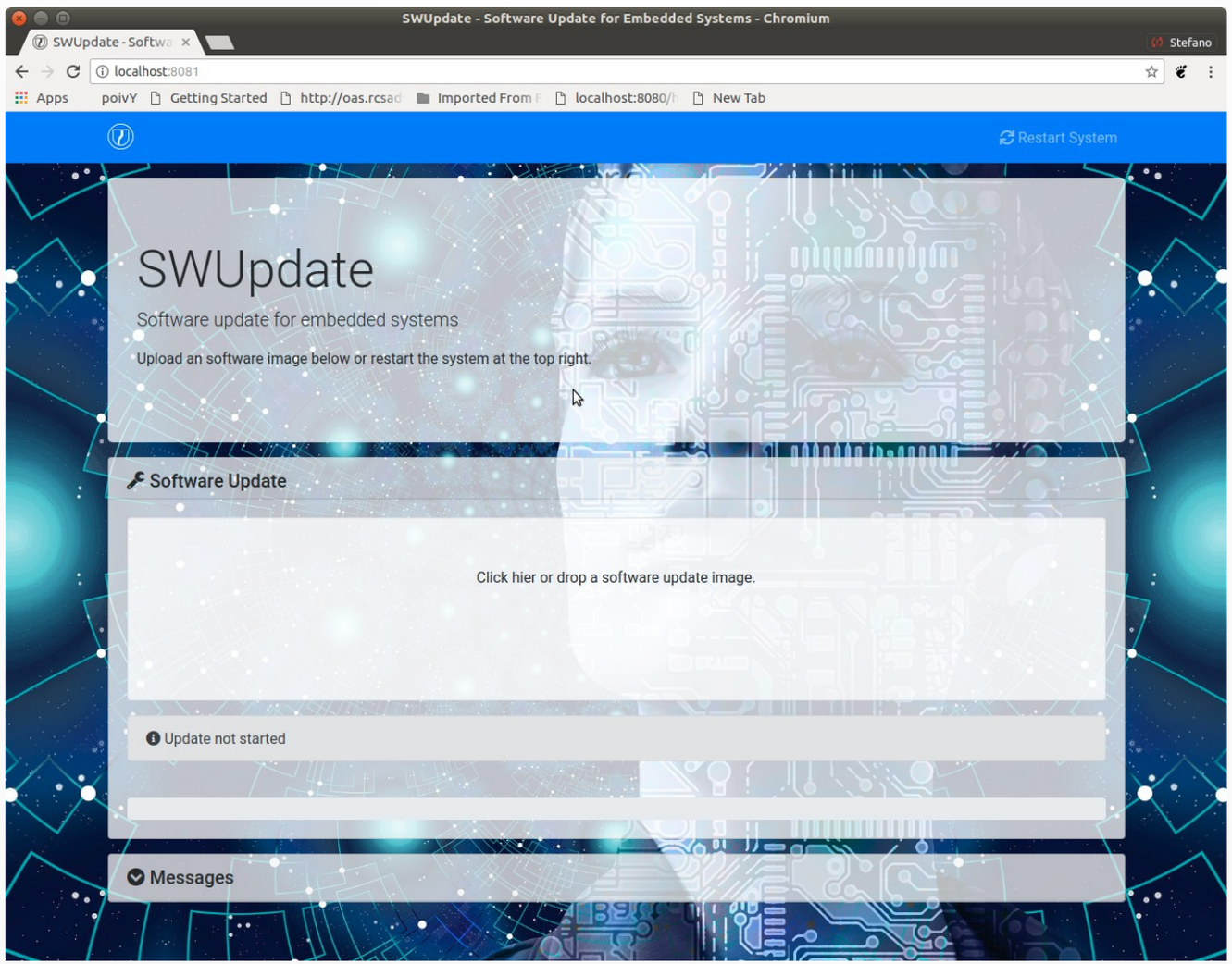
- system-update running on GW
- SWUpdate : the usual updater (single device)
- Pull-update: load SWU during HW substitution

Device SWU

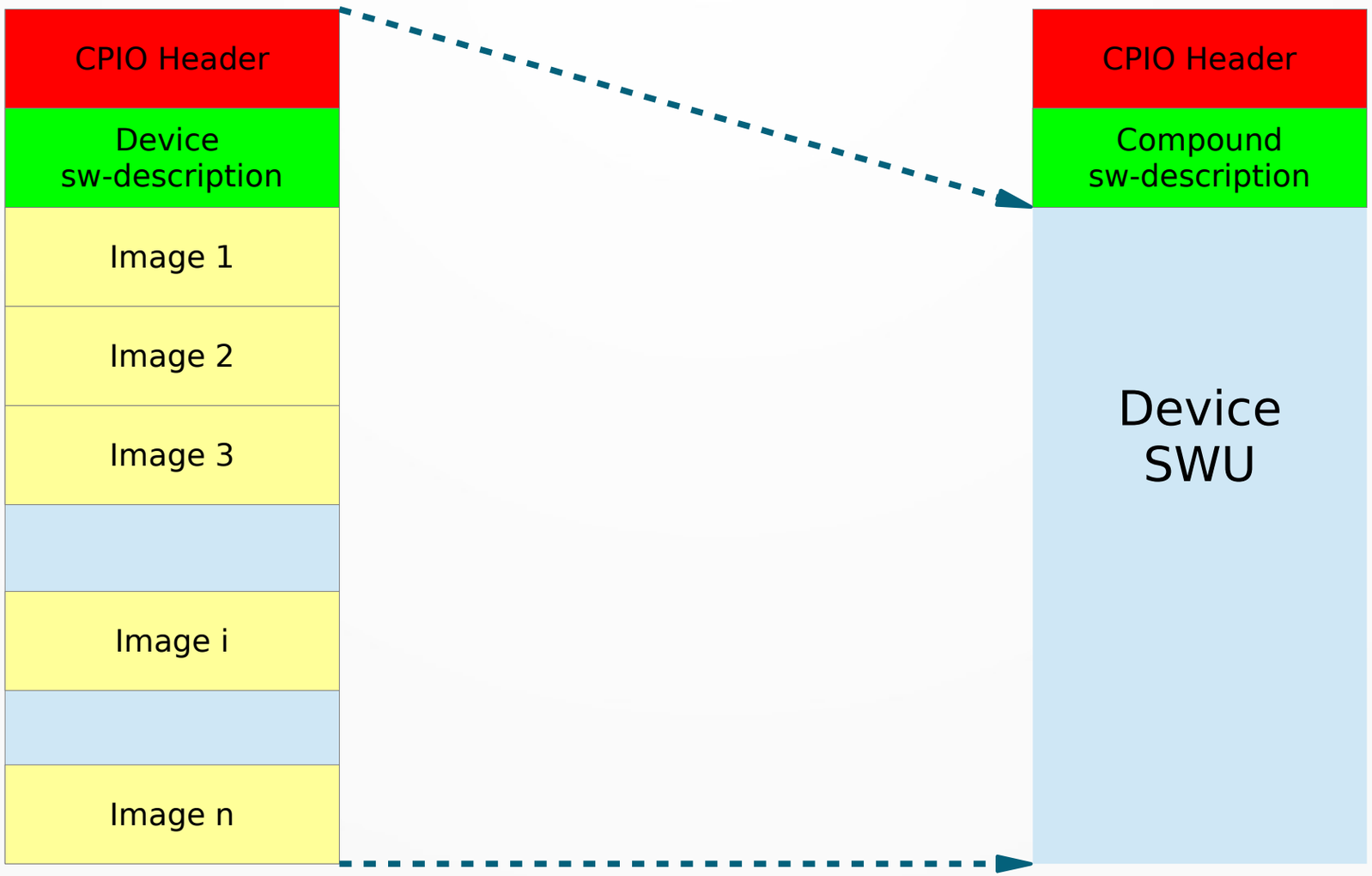


```
software = {
    device-controller = {
        hardware-compatibility: [ "1.0"];
        rescue : {
                partitions: ( /* ubi volumes */  );
                images: (   {.....});
                uboot: ( {.....}  );
            };
        };
        production : {
            copy1 : {
                    images: (...)
                    uboot: (...)
            };
            copy2 : {
                    images: (...)
                    uboot: (...)
```

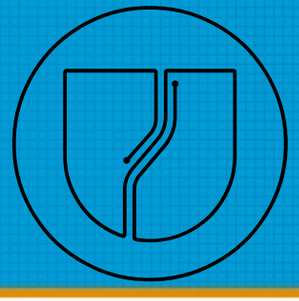
SWUpdate's web interface



Device SWU as payload



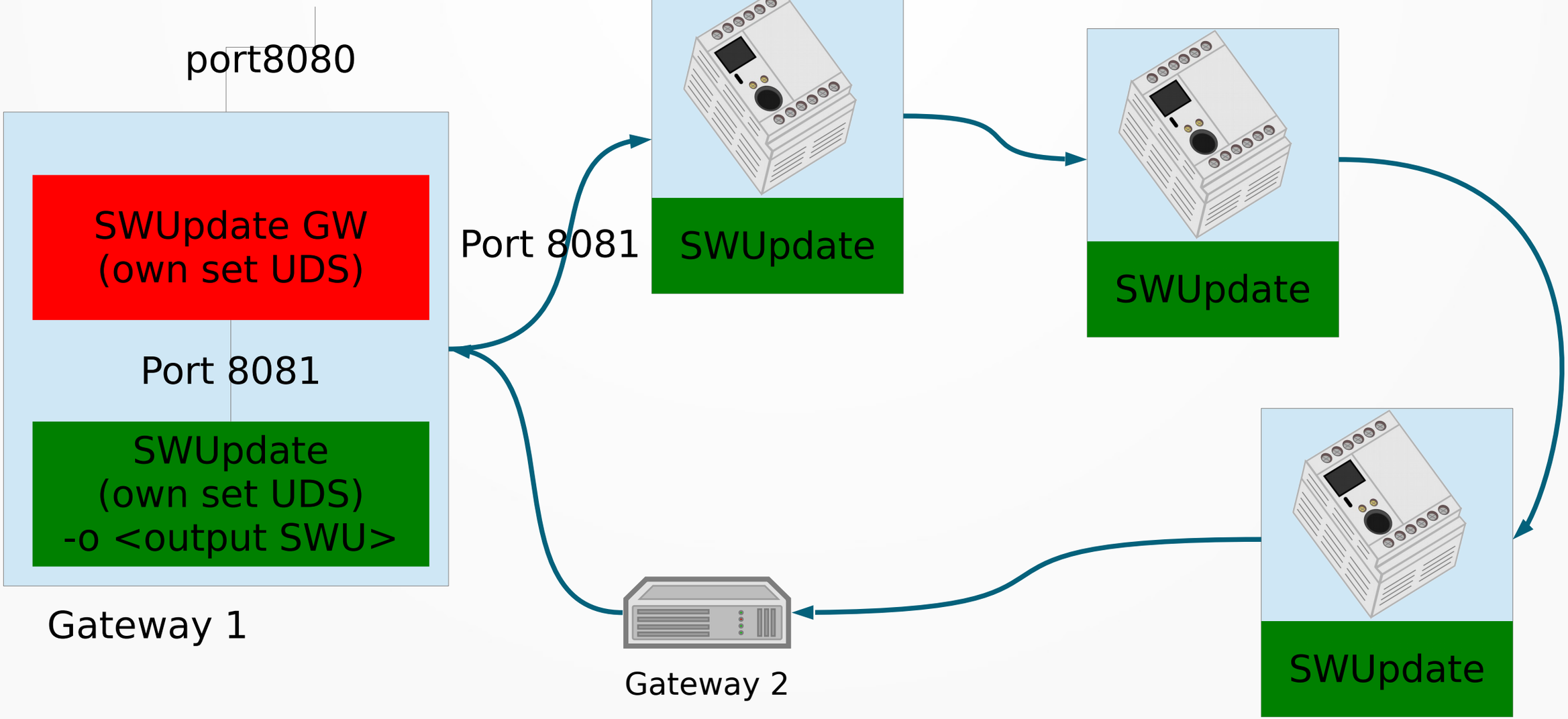
Compound image



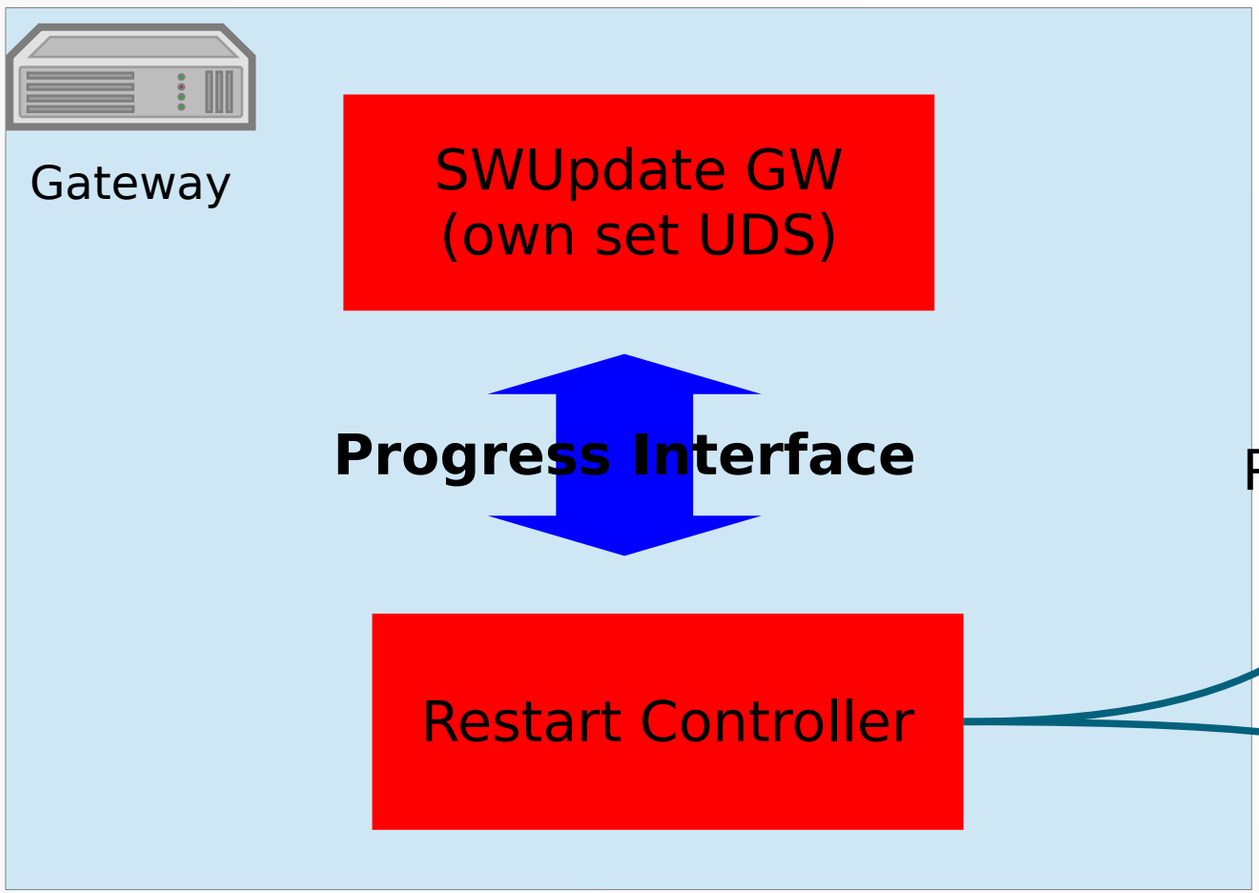
```
software = {
    gateway-controller = {

    embedded-script = „
            function detect_topology(image)
                ……….
            end
        hardware-compatibility: [ "1.0"];
        images: (
                    {
                        filename = "<SWU Image for each device>";
                        type = "swuforwarder“;
                        sha256 = "<hash 256 of SWU>“;
                        hook = "detect_topology“
                        properties {
                        }; // this will be filles by the embedded script
                    };
            };
```

Functional behaviour

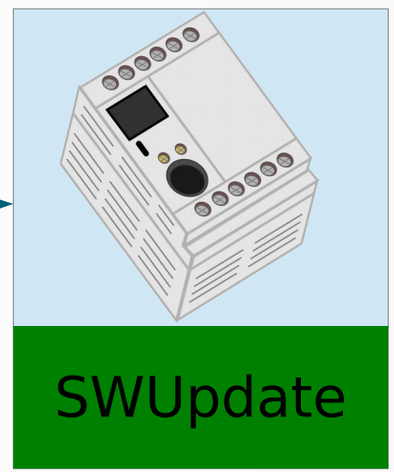


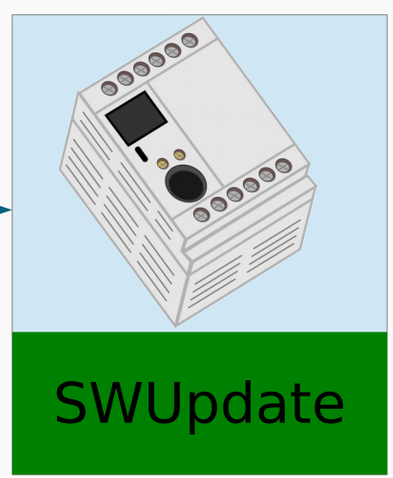
System restart



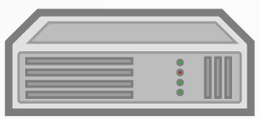
POST /reboot

POST /reboot

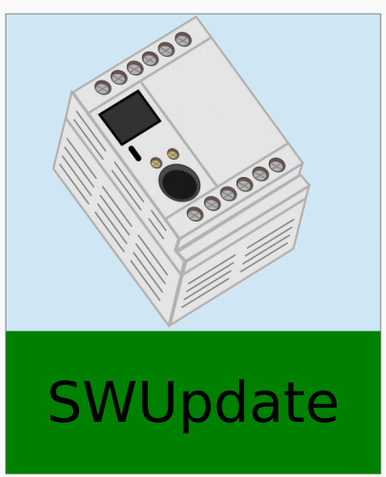




SW sync at boot



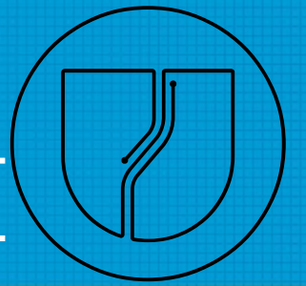
Gateway



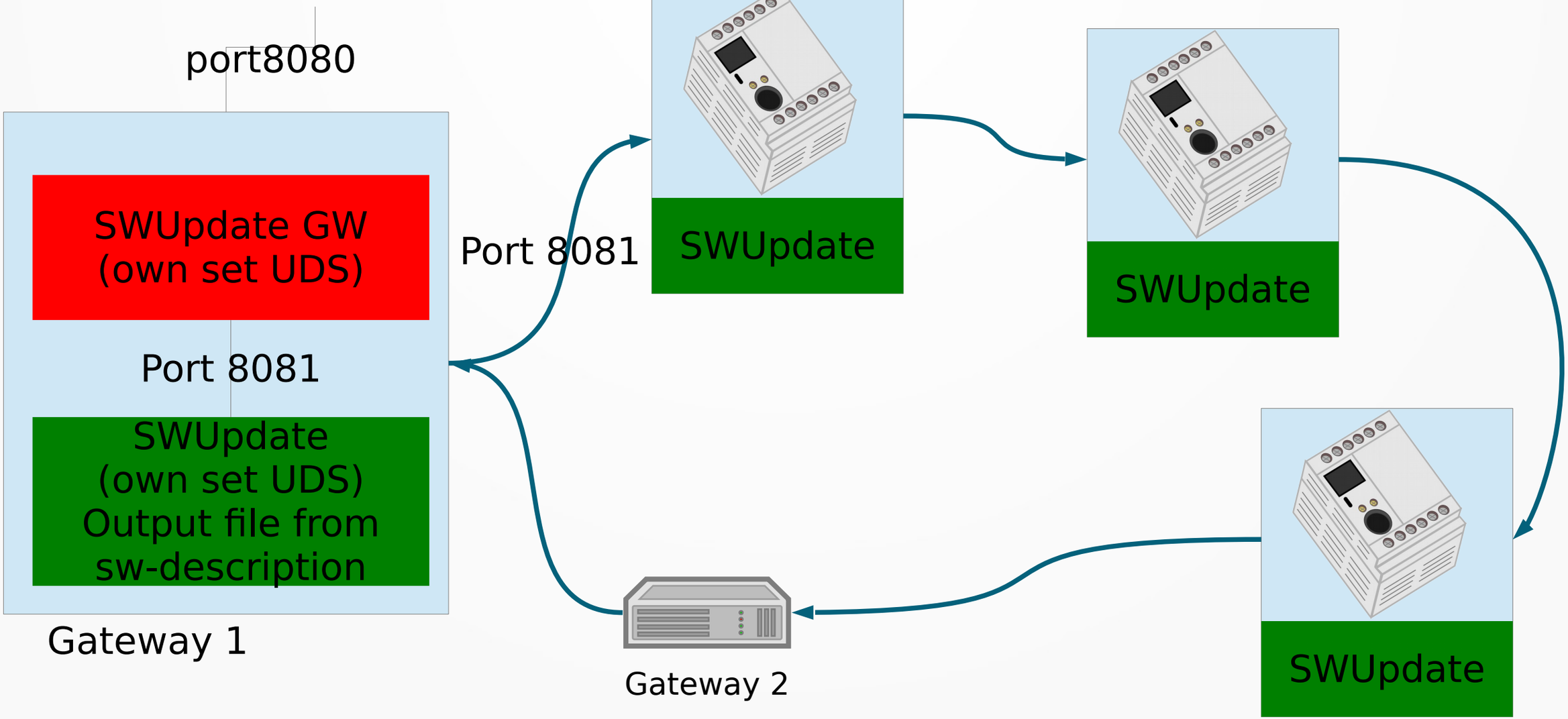
SWUpdate

- Device Check if SWU is available
- Download first Kbs
- SWUpdate sw-description, extracts version=x.y.z
- If version differs, run swupdate in dry mode
  - Swupdate -v -n -d „-u <gw ip>“
- If last work, update from GW

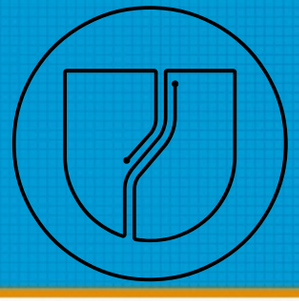
Configuration deployment



Deploy: CFG as SW



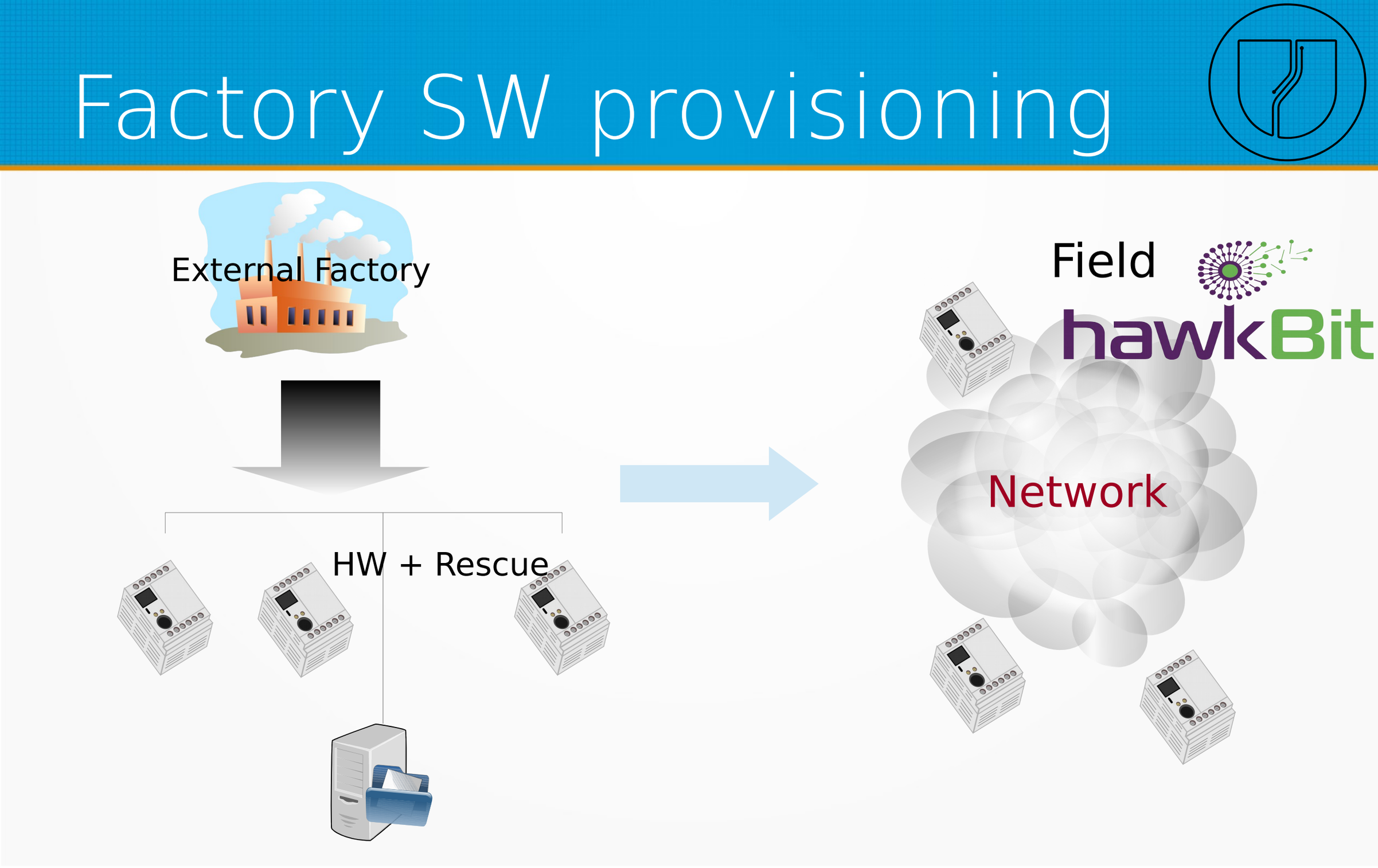
Configuration SWU



```
software = {
    device-controller = {
        output = „config.swu“
        hardware-compatibility: [ "1.0"];
            production : {
            copy1 : {
                    files: (
                        filename = „configuration.tar.gz“
                        type = „archive“;
                        compressed = true;
                        path = „/etc/application“;
                    );
            };
            copy2 : {
                    ref = „ ../copy1“;
            }
```

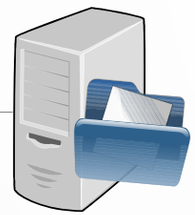
Factory SW deployment

Factory SW provisioning



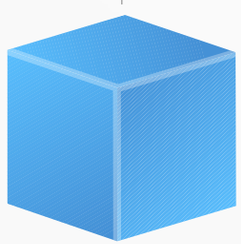
Updater Proxy

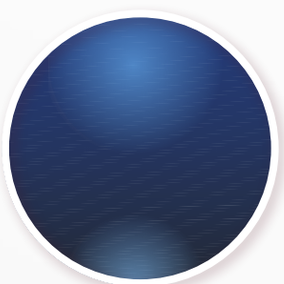
Proxy for small devices



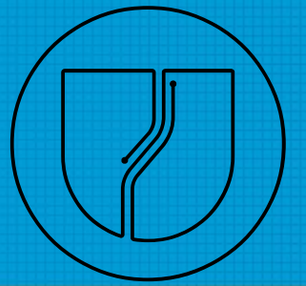
hawkBit

Updater Proxy

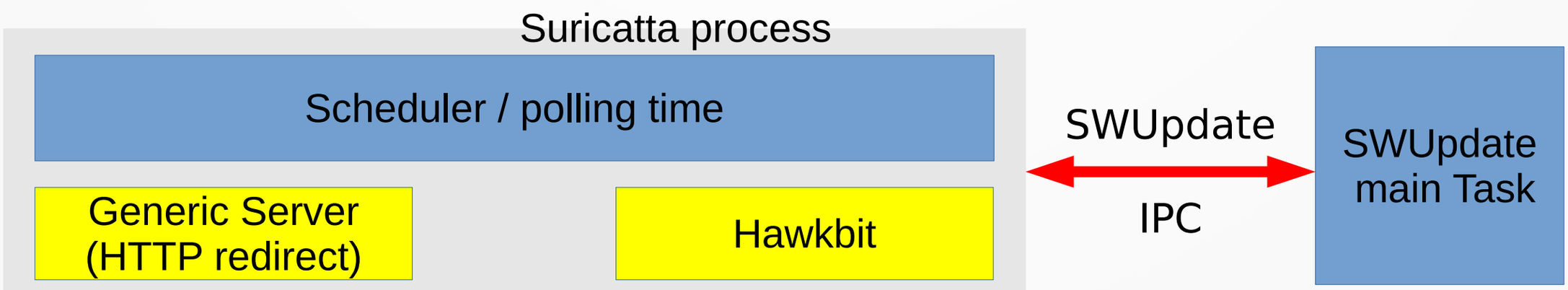




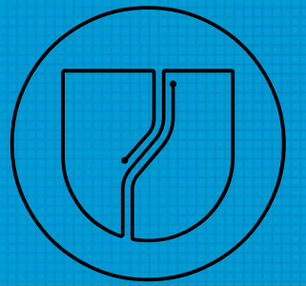
Server unawareness



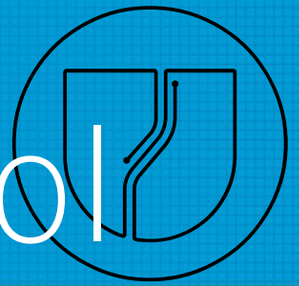
Layer to the server



Custom protocol

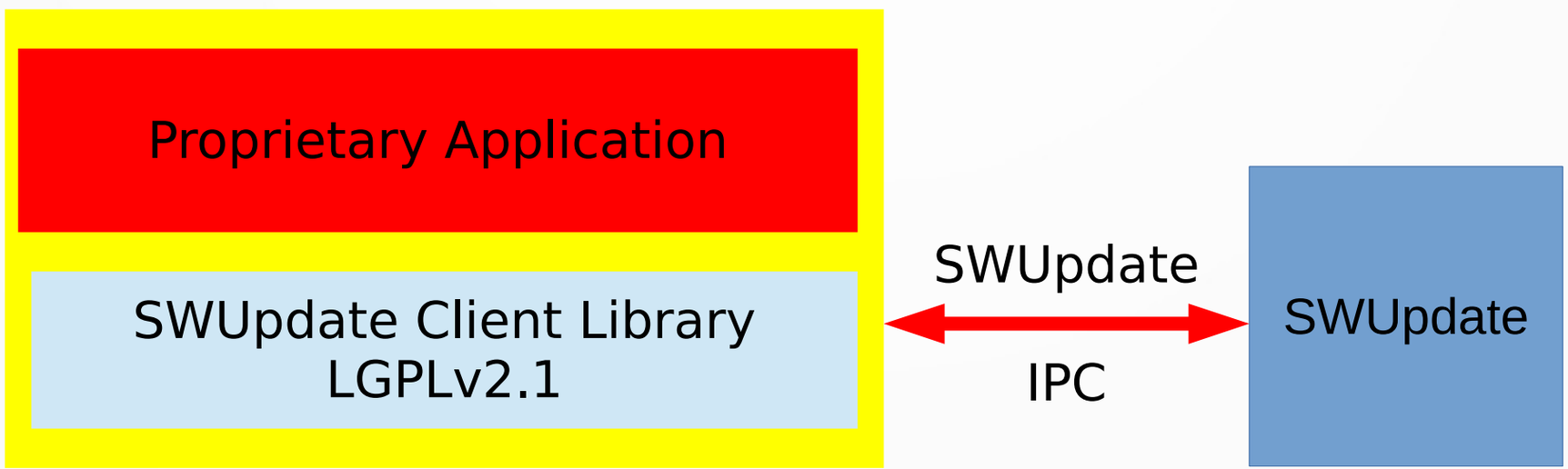


Reason for custom protocol

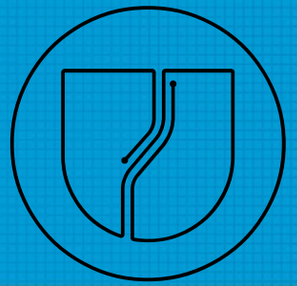


- There is already a well defined standard
- Project has already an own download method
- Compatibility with past / previous device generation

Bind to SWUpdate

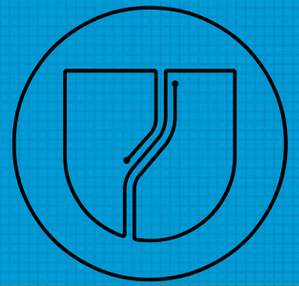


Feature request: selective downloading



- EV Charging business
- Protocol standardised (OCPP)
- Vendors have many variants of devices and mode of operations
- Admins of backend just manage update files on vendor basis, not on devices
- Request for one file, but the updater should retrieve just the parts that must be installed.

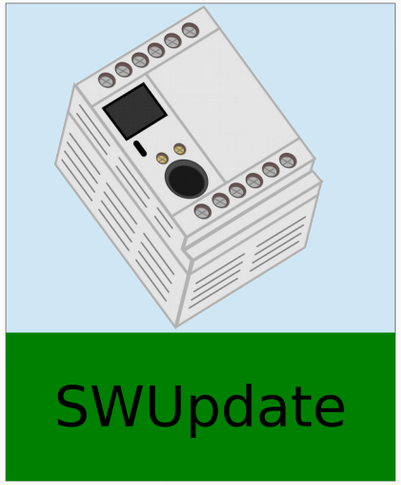
Continuous SW development



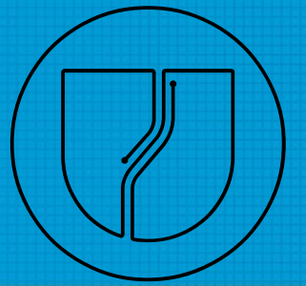
- SW is installed on device for all developers
- Single developer does not need to bother with update
- Single way to update, closer to the case in field
  - An update process is well tested before production
  - Ensure developers are working with same SW

Automatic delivery - CI

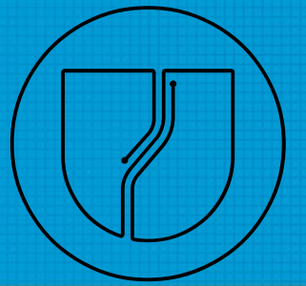
https://github.com/Rahix/tbot



Integration with buildsystems



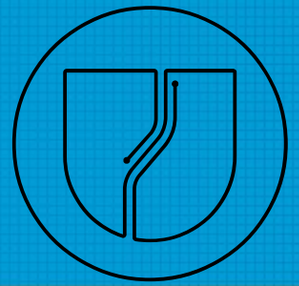
More way to build



yocto PROJECT

isar

Summary



- Not just update „standard“ artifacts
  - Flash, FPGA, uC, etc.
- Deploy not just SW
  - Configuration, OEM Data, etc.
- Update complex system as one single device
- Support different build systems
- Support different fleet deployment servers

swupdate@googlegroups.com