Meet an all scenarios os:
a distributed OS with feet on the ground

Davide Ricci
Open Source Technology Center, Huawei
Disclaimer

Don’t get too attached to “an all scenarios os” nickname because it is temporary and will soon be gone
(thanks, you have served us well “an all scenarios os” but it’s time to move on)

New project name will be announced during EclipseCon 2021 – 25th to 28th of October

as a follow up to recent announcements of collaboration between the Eclipse Foundation and the Open
Atom Foundation.
IoT - Problem Statement

- Dumb
- Technological fragmentation
- Reinventing the o.s. wheel
- Lack of interoperability at the edge
- Partial, brand-centric interoperability
- Cloud centric: compute, interoperability
- Top-down, cloud providers driven
- Inefficient, insecure, expensive
**IoT - Problem Statement**

**CONSUMERS**
- Complexity
- Insecurity
- Lack of privacy
- Turned into products

**DEVICE MAKERS/OEMs**
- Reinventing the wheel
- Sub-optimal choices
- Becoming Device Dealers
- Monetize consumer’s data

**CONTENT CREATORS**
- Lack of choice
- Lack of standards
- Drive o.s. / cloud stickiness
- Influence device makers
- Monetize consumer’s data
Smart Things

- Smart lights, motion sensors
- Smart door locks
- Smart thermostats, radiators, valves
- Smart cameras, doorbells, alarms
- Smart TVs, projectors, speakers
- Smart wearables

- Sensors and actuators
- MCU, CPU
- FreeRTOS, Zephyr, LiteOS, Linux,...
- From KBs to GBs
- W or w/o display (simple graphics)
- Zigbee, BT,... close range comm
- Java, JS, C, C++ apps
Smart Gateways

- Things to things, things to cloud comms
- Brand specific / isolated comms
- Compute / Storage / OTA / Comms
- CPUs
- Linux, headless
- Gbs
- WiFi, Eth, Zigbee, BT, ...close to medium range comm
Smart Mobile

- Phones, Tablets, TVs,...
- Smart Things configuration
- Brand specific smart things apps
- Compute / Storage / HMI / Trainers
- CPUs, Gbs
- Linux, accelerated graphics, rich display
- Medium to long range comms
# Putting it all together

## Distributed Functionalities

<table>
<thead>
<tr>
<th>Distributed Communications</th>
<th>Distributed Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actuator</td>
<td>Actuator</td>
</tr>
<tr>
<td>Actuator</td>
<td>Actuator</td>
</tr>
<tr>
<td>Actuator</td>
<td>Actuator</td>
</tr>
<tr>
<td>Actuator</td>
<td>Dist. HMI</td>
</tr>
<tr>
<td>Dist. HMI</td>
<td>Dist. HMI</td>
</tr>
</tbody>
</table>

### Distributed Compute and Storage

- Autonomous Agents

## Applications

- Application Framework
- Applications

## Device Functionalities

<table>
<thead>
<tr>
<th>Device Performance</th>
<th>Devices</th>
<th>Gateway</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Range - meters</td>
<td>10</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>Energy Consumption</td>
<td>uWatts</td>
<td>mWatts</td>
<td>Watts</td>
</tr>
<tr>
<td>Memory footprint</td>
<td>kB</td>
<td>MB</td>
<td>GB</td>
</tr>
<tr>
<td>Processor speed - MIPS</td>
<td>100</td>
<td>500</td>
<td>1000</td>
</tr>
</tbody>
</table>

- Phone, Maps, Location, ...
- Autonomous Agents and Orchestrators

### Device Performance

- Javascript / C-C++ /Java
- Javascript / C-C++
- GN, Jav, ...

### Device Type

- Devices
- Speakers, Earbud, Light Bulbs, Doorlocks, Appliances, Watches, Thermostats,...
- Transparent GWs
- Phones, Tables, In-car
Mission Statement

- Open Source
- Open Governance
- Industry Driven
- Interoperable (cross-brand)
- User Centric (experience, privacy, security)
- Distributed edge o.s.
  (the cloud is “just” a citizen, not the king)
Based on the W3C semantic web and IEEE distributed agents works

- Distributed agency:
  - Agents have different physical characteristics
  - Agents can play a different role in different scenarios
  - Agents broadcast their characteristics
  - Agents elect one agency coordinator
  - Coordinators trigger coordinated execution, recruit available agents, distribute tasks
Ontologies are used to describe:
- Devices characteristics
- Problems, tasks, routines

Agency coordinator:
- Matches agents ontologies with problems
- Select agents, distribute tasks, executes
- Agency training is supervised or unsupervised
African Lion Ontology:

Classes: African Lion --> Lion --> Animal

Class: Book About Animals

Instances: actual book about African lions and its properties
Edge Devices Collaboration

Autonomous agents

Things: sensors, actuators, compute

Gateways: compute, storage, communication, coordination

Mobile: compute, storage, HMI, agency supervised training, coordination (not ideal, back-up)
## Putting it all together

### Distributed Functionalities
- Discovery
- Sensor
- Actuator
- HMI
- Distributed compute and storage
- Edge AI

### Distributed Communications
- Actuator
- HMI
- Autonomous Agents
- Dist. HMI
- Dist. Compute and Storage
- Autonomous Agents and Orchestrators

### Applications
- Application Framework
- Applications: Phone, Maps, Location, ...

### Device Functionalities
- Kernel: Zephyr/LiteOS, Linux
- CPU type: MCU, CPU
- Number of CPUs: 1, 2, 2, 4, 8
- Display: Headless, Display, Headless, Display
- GPU Acceleration: Simple Graphics, Accelerated Graphics
- Application Runtime Engine: Javascript / C-C++, Javascript / C-C++ / Java
- Application Framework: GN, Jav, ...

### Device Performance
- Communication Range - meters: 10, 100, 1000
- Energy Consumption - uWatts, mWatts, Watts
- Memory footprint - kB, MB, GB
- Processor speed - MIPS: 100, 300, 1000

### Device Type
- Devices: Speakers, Earbud, Light Bulbs, Doorlocks, Appliances, Watches, Thermostats, ...
- Things
- Gateway: Transparent GWs
- Mobile: Phones, Tables, In-car
Some relevant layers

- meta-ohos: root layer
- meta-openembedded
- meta-clang
- meta-zephyr, meta-freertos, meta-liteos
- meta-riscv
- meta-ohos: openharmony components
- meta-seco, meta-st, meta-av96, meta-intel, ...
Build flavours and supported HW

- Build flavours = supported kernels

- Currently supported: Zephyr, Linux
  
  WIP: LiteOS, FreeRTOS

- Supported images / machines:
  
  - Linux: allscenarios-image-base (headless), allscenarios-image-extra
  
  - Linux: qemu86-64, qemu86, qemuarm, qemuarm64, seco-intel-b68 (SECO SBC-B68), stm32mp1-av96 (96Boards Avenger96), seco-imx8mm-c61 (SECO SBC-C61), raspberrypi4-64 (Raspberry Pi 4 Mobel B)

  - Zephyr: zephyr samples
  
  - Zephyr: qemu-x86, qemu-cortex-m3, 96b-nitrogen (96Boards Nitrogen), 96b-avenger96 (96Boards Avenger96), arduino-nano-33-ble (Arduino Nano 33 BLE and Arduino Nano 33 BLE Sense), nrf52840dk-nrf52840 (Nordic Semiconductor nRF 52840 Development Kit)
Blueprints

- Minimum viable, 80% production-ready reference solutions
- Pre-integrated, tested and maintained (LTS)
- Both makers and production silicon
- Enable and test cooperative use-cases
- Blueprints are NOT: full featured, optimized for cost, size, pretty looking
- Expect to see boards, wires, etc.
- Available blueprints: Doorlock, Transparent Gateway, Touch Panel
- WIP blueprints: Vending machine, mobile phone, smart speaker with vocal assistant, robotic companion,...
Linux kernel hardening options:

- Memory allocator
- Disabling apparently useful but obsolete features such as COMPAT_BRK, PROC_KCORE, BINfmt_MISCSECURITY_DMESG_RESTRICT
- Compiler level hardening via FORTIFY_SOURCE
- Disable physical memory access and detect unsafe memory permission
- Hardened usercopy from userspace
- Kernel structures data validation
- Under consideration: IOMMU, Panic on Oops

Security related layers / components:

- meta-security
- meta-security-compliance
- meta-security-isafw
- meta-tpm
High level architecture
Continuous Integration

- gitlab runners for builds, with git-repo cache, bitbake sstate and download cache
- strategic placement of jobs across repositories to ease maintenance
- lava for smoke testing on hardware and in virtual environments
- Scancode, Fossology, REUSE, Debian matcher for license compliance and SPDX SBOM
- extra care for fork based workflow in multi-repo world and bitbake recipie (revision pinning) world
Continuous Integration

- Shared jobs for images (machine / flavours) officially supported:
  - Currently 14
  - Linux-*, zephyr-*, freertos-armv5, blueprints-*
- Hidden jobs as foundation building blocks that shared jobs leverage:
  - workspace --> assembles all repos via git repo
  - bitbake-workspace --> initialized bitbake build
  - build-linux, build-zephyr, build-freertos, build-liteos
  - build-recipe, build-image
  - build-docs
  - lava-test, lava-report
  - ip-scan
Decentralized, distributed device testing
Each member, contributor,... can add physical devices at different locations
Device added under testing can be shared via public cloud infrastructure
Each site can add one to hundreds of devices
Sites broadcast their availability to a central repository / directory
LAVA (Linaro Automation and Validation Architecture) is used:
lava-test calls LAVA and create a testing job
lava-report iterates through the active jobs collects results and aggregates them in a report
OpenHarmony application compatibility

- OpenHarmony is an OpenAtom foundation project
- Unified ecosystem is achieved by building “OpenHarmony compatible” o.s. images
- Compatibility is defined in a Compatibility Specification and automated via an Application Compatibility Test Suite (ACTS)
- xts_acts jobs / testing (https://git.ostc-eu.org/OSTC/OHOS/components/staging/xts_acts)
- OpenHarmony components needed to achieve compatibility packaged in meta-openharmony-x.y.z
- GN class for bitbake in order to build native GN openharmony files
Openchain Specification 2.0 conformant


Training, R&R, fundings, activity, IP auditing embedded into R&D

Continuous IP compliance via integrations of IP compliance toolchain in the dev process via gitlab jobs

Low Resolution SBOM: Merge --> Scan --> SPDX / BOM --> Dashboard

High Resolution: Dashboard --> IP Auditor --> Fossology

Releases SBOMs for alpha, beta and official yearly release
Maintenance and Release life cycle

- One yearly major release (12 months dev cycle)
- 3 years LTS with decrease level of service (based on bug / CVE impact score)
- Dedicated LTS team: maintenance engineers, security response engineers
- Leverage upstream LTS for major components such as linux kernel, toolchain, ...
Maintenance and Release life cycle

Jasmine 12.2021
- Development
- LTS y.1
- LTS y.2
- LTS y.3

Goofy 12.2022
- Development
- LTS y.1
- LTS y.2
- LTS y.3

Daisy 12.2023
- Development
- LTS y.1
- LTS y.2
- LTS y.3

YEAR 1
- BUGS: All
- CVEs: All

YEAR 2
- BUGS: P1/P2
- CVEs: CVSS 7-10

YEAR 3
- BUGS: P1
- CVEs: CVSS 9-10
Project Phases

01 Bootstrap – 12 to 18 months
- New brand
- Hosting foundation
- Growing active members, design wins and community

02 De-fragmentation (broad sustainable industry participation) – 12 to 36 months

03 Adoption (design wins, shipped devices, apps and content) – 18+ months

04 Edge Devices Collaboration – 18+ months
Thank you