OPEN CONNECTIVITY FOUNDATION®

IoTivity Overview

OCF EU Developer Training 2019, Budapest, Hungary

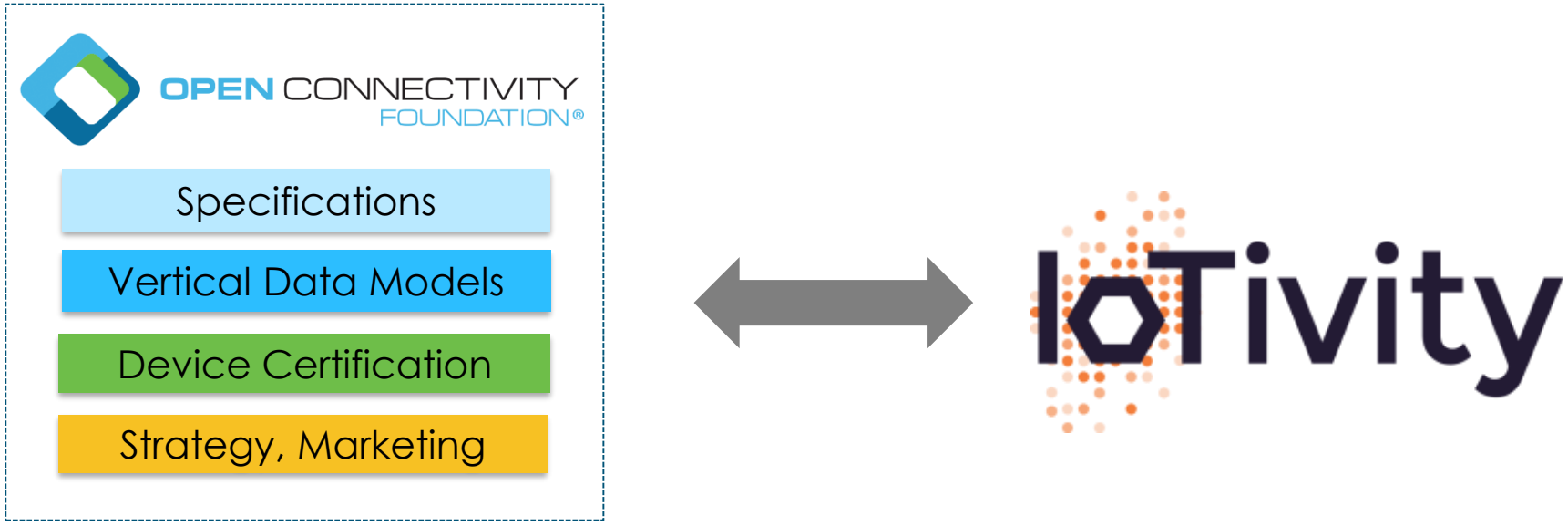
Kishen Maloor, Intel

Outline

- What is IoTivity?
- Structure of an OCF implementation
- IoTivity-Lite
  - Protocols and payloads
  - Support for OCF roles
  - Porting
  - Directory structure
- IoTivity-Lite resources

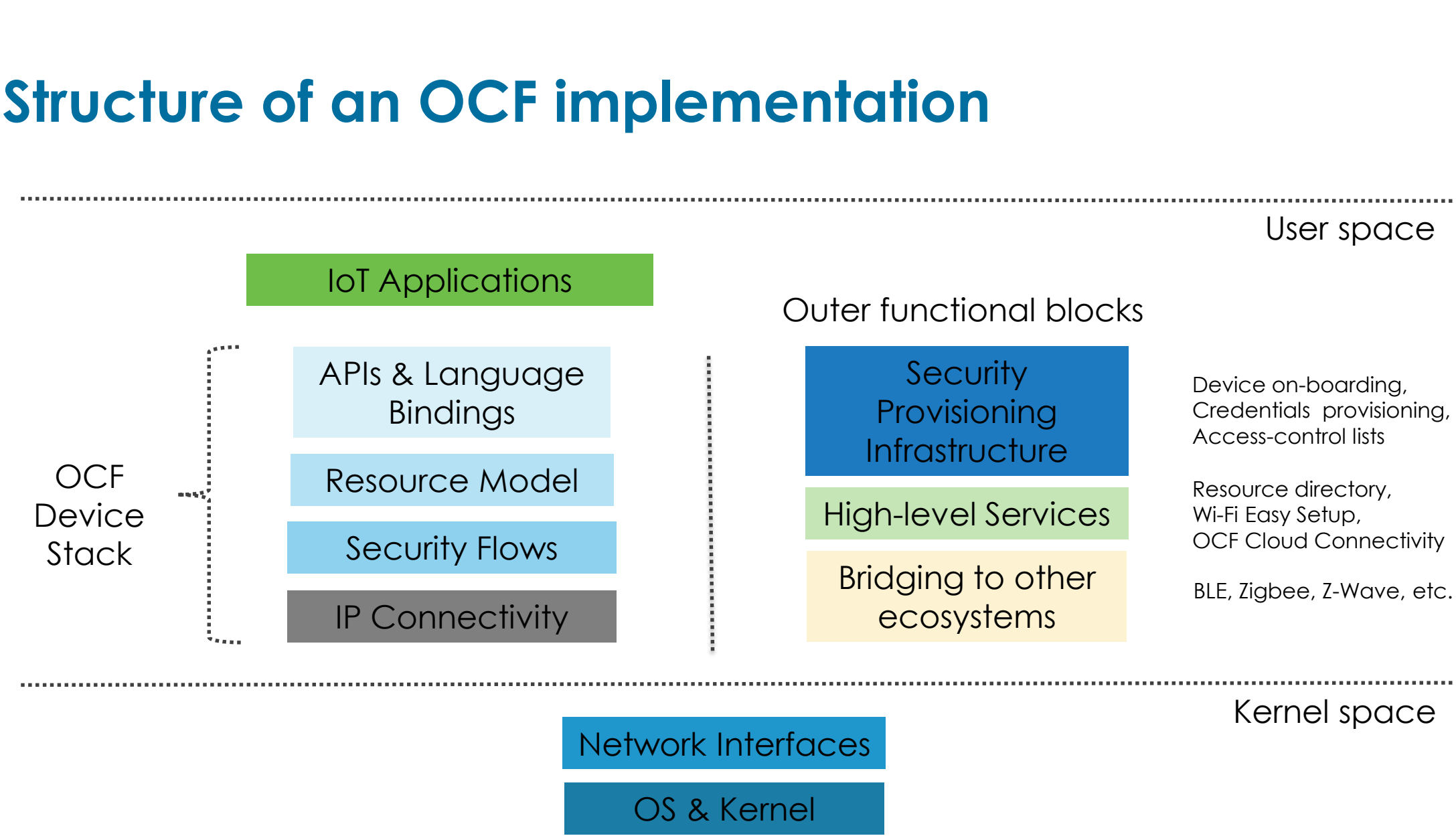
What is IoTivity?

- Umbrella of projects for building IoT devices
- Open-source, reference implementations of OCF specifications
- Serve as starting point for developing and certifying OCF products



Independent governance with coordinated efforts

Structure of an OCF implementation



IoTivity-Lite

- Lightweight implementation of OCF specifications
- Suitable for all device classes (including few constrained devices)
- Port to any target by implementing a thin platform abstraction layer
- Runs on Linux, Windows, Android[1], macOS[2], and multiple RTOSes
- C and Java[3] APIs

1, 3: Working Android adaptation with Java binding currently on "swig" branch

2: Work-in-progress

3: Java bindings may be used to build Java applications for platforms with the Java runtime (Eg. Linux, Windows, etc.)

Protocols and payloads

- Constrained Application Protocol (RFC 7252)
  - Lightweight protocol for constrained nodes and networks
- Security
  - DTLS-based authentication, encryption and access control
  - Leverages mbedTLS https://github.com/ARMmbed/mbedtls
- Concise Binary Object Representation (RFC 7049)
  - Handle OCF request/response payloads using simple C APIs
  - Payloads typically consist of key-value pairs
  - Leverages tinyCBOR https://github.com/intel/tinycbor

Support for OCF roles

- REST architectural style; "things" modeled as resources
- Servers
  - Expose resources to Clients
- Clients
  - Access resources hosted in Servers
- Onboarding Tools (OBT)
  - Takes on the Client role
  - Manage security context across a network of OCF Devices
  - APIs for creating OBTs

Porting

- OS-agnostic core
- Abstract interfaces hook into platform-specific components
- Bounded definitions, elicit specific contract from implementations
- Platform-specific blocks
  - Clock
  - Connectivity
  - PRNG
  - Storage

Directory structure – <IoTivity-Lite root>/*

api

include

messaging

port

tests

tools

LICENSE.md

README.rst

apps

onboarding_tool

deps

security

patches

util

IoTivityConstrained-Arch.png

Directory structure - <IoTivity-Lite root>/port/*

oc_connectivity.h

oc_random.h

oc_log.h

oc_assert.h

linux

android

...

oc_clock.h

oc_storage.h

oc_network_events_mutex.h

windows

zephyr

IoTivity-Lite resources

- IoTivity-Lite repository
  - https://github.com/iotivity/iotivity-lite
- IoTivity-Lite build instructions
  - https://github.com/iotivity/iotivity-lite/blob/master/README.rst
  - Each OS adaptation (port) employs a build system native to its environment (E.g. Linux uses make, Windows uses VS projects, etc.)
- IoTivity Wiki
  - https://wiki.iotivity.org/
- OCF Specification documents
  - https://openconnectivity.org/developer/specifications

OPEN CONNECTIVITY FOUNDATION®