OCPP2.lite

Plugfest February 2024

www.OpenChargeAlliance.org

Why OCPP2.lite?

We are adding a lot of features to OCPP for important market segments

But….

At the same time, there are many booming markets for lightweight – and low cost - OCPP implementations

That is why we are turning our attention to OCPP2.Lite!

A booming market: 2 wheelers in India

75 Million e2w in 2030

10.5 Million e4w in 2030

EV adoption will be led by 2 wheelers in India with a 7:1 ratio

Projections by Niti Aayog – The Indian Think Tank

Economies if everything is standardized based on e4w

Avg Battery Size	40 kWh	4 kWh	
Avg Cost of Vehicle	$30,000	$1,200	
% added to BOM by connector @ $100	0.33%	8.3%	Hardware
Cost per charge cycle 0-80%	$8.5	$0.85	
Cost of communication for charging @ $0.04	0.5%	5%	Software

Details shared are based on vehicles in India, cost of charging is set as $0.26 per unit of electricity

What does a market need regarding data communication? And can we get there with OCPP2.lite?

Mobile networks are an important cost factor in Kenya

Kenya on the world map[1]

Typical mobile internet costs:
Flatrate, 50 MB: USD 0.7
Flatrate: 200 MB: USD 1.0

The case of CHAJI[2], a peer-to-peer charger sharing network in Kenya
Private persons operate chargers and provide access to everyone via OCPP. Mobile connectivity is a must in rural areas. USD 0.7 is considered a reasonable monthly operational cost.

Target mark for OCPP
To be competitive as a standard, it should be possible to operate a charger via OCPP with **at most 25 MB per month** (50% headroom).

1) CC BY-SA 3.0 Deed https://de.wikipedia.org/wiki/Datei:Kenya_on_the_globe_(Africa_centered).svg
2) https://www.chajigo.com/

We have started a new Task Group that meets every 3 weeks

Track 1

What do we already have within 2.0.1

Track 2

What can we do more within 2.x

Track 3
What is more like 3.x
(will not work with 2.x)

Track 1: What do we already have within 2.0.1?

Lightweight Functionality
- Description of minimal implementations in the 2.0.1 spec
  - Part 0 par.4 describes the basic implementation of 2.0.1
  - Part 1 par.4.6 describes the minimum Device Model
  - Part 5 par. 3.1 summarizes the optional features
- There is also a special 'Mode ½ only' certification profile

Transport layer
- Websocket compression to minimize data
- Charging stations can select only ECC keys for the cert chain and have a maximum chain length for verification

Light(er)weight Code
- Code in C++, e.g. Open Source stack MicroOCPP

First identified Task Group Actions

1) Write a Whitepaper on how to implement OCPP2.0.1 as 'Lightweight'

2) Gather benchmark metrics for OCPP implementations, both 1.6 and 2.0.1
   - Based on implementations by MicroOCPP and others
   - Metrics for RAM, Flash memory, data communication, Network Round trip time, …

3) Compare the benchmark OCPP2.0.1 'Lightweight' metrics with the Target Metrics, based on Industry needs

Track 2 and 3: What can we do more?

Track 2: What can we do more within 2.x

- ➢ Transport Layer Optimization:
  - Maximum fragment length negotiation?
- ➢ Improved Commissioning Flow
  - OCPP could also define a standard commissioning interface:
  - standard URL, QR code, REST API, BLE service, NFC packet, etc.
  - Get this from your OCPP CSMS and easily send it to your EVSE
- ➢ OCPP Hub: Low resource EVSEs can often only support a single OCPP connection. Maybe we define a standard for an OCPP Hub?
- ➢ Operation type "ReEnroll": After boot, the charger can send ReEnroll to the backend which then sends all persistent data to the charger again (i.e. ChargingSchedules, Variables, Reservation, Availability). The charger only needs to store the backend URL and certificates.
- ➢ "Transaction resumption": After ReEnroll, the backend could resume ongoing transactions by sending a RemoteStartTransaction request.
- ➢ Provide a memory footprint evaluation tool for new OCPP features (across the working groups)

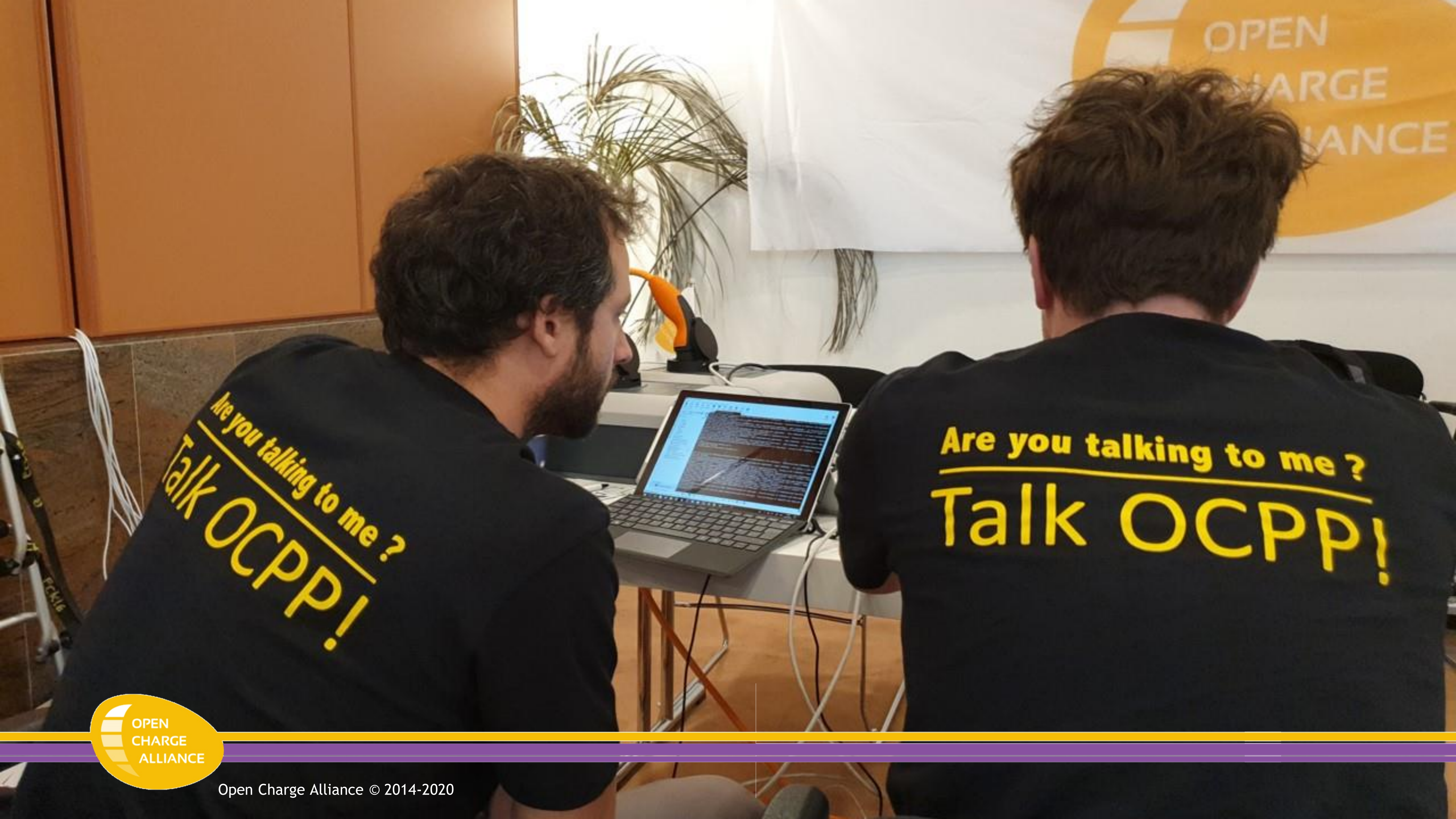
Track 3: What is more like 3.x (will not work with 2.x)

- ➢ Transport Layer Optimization:
  - Use raw public keys in trust store instead of certs
  - Use DTLS/CoAP

- ➢ Message Encoding Optimizations: move to a more efficient encoding for machines:
  - MessagePack: Designed to be transparently converted to/from JSON; Larger than Protobuf messages, but maybe faster to serialize/deserialize (https://medium.com/@hugovs/the-need-for-speed-experimenting-with-messageserialization-93d7562b16e4)
  - Protobuf: ~30% the size of JSON and 6x faster to pr [illegible] (https://nilsmagnus.github.io/post/prot [illegible] self-describing.

Work in Progress: join the Task Group!

OPEN CHARGE ALLIANCE


OPEN
CHARGE
ALLIANCE
Are you talking to me ?
Talk OCPP!
Are you talking to me ?
Talk OCPP!