Project Kona

Darryl Mocek
Principal Member of Technical Staff
Oracle Corporation

Zach Shelby
Vice President, Marketing
ARM Internet of Things BU

October 27, 2015

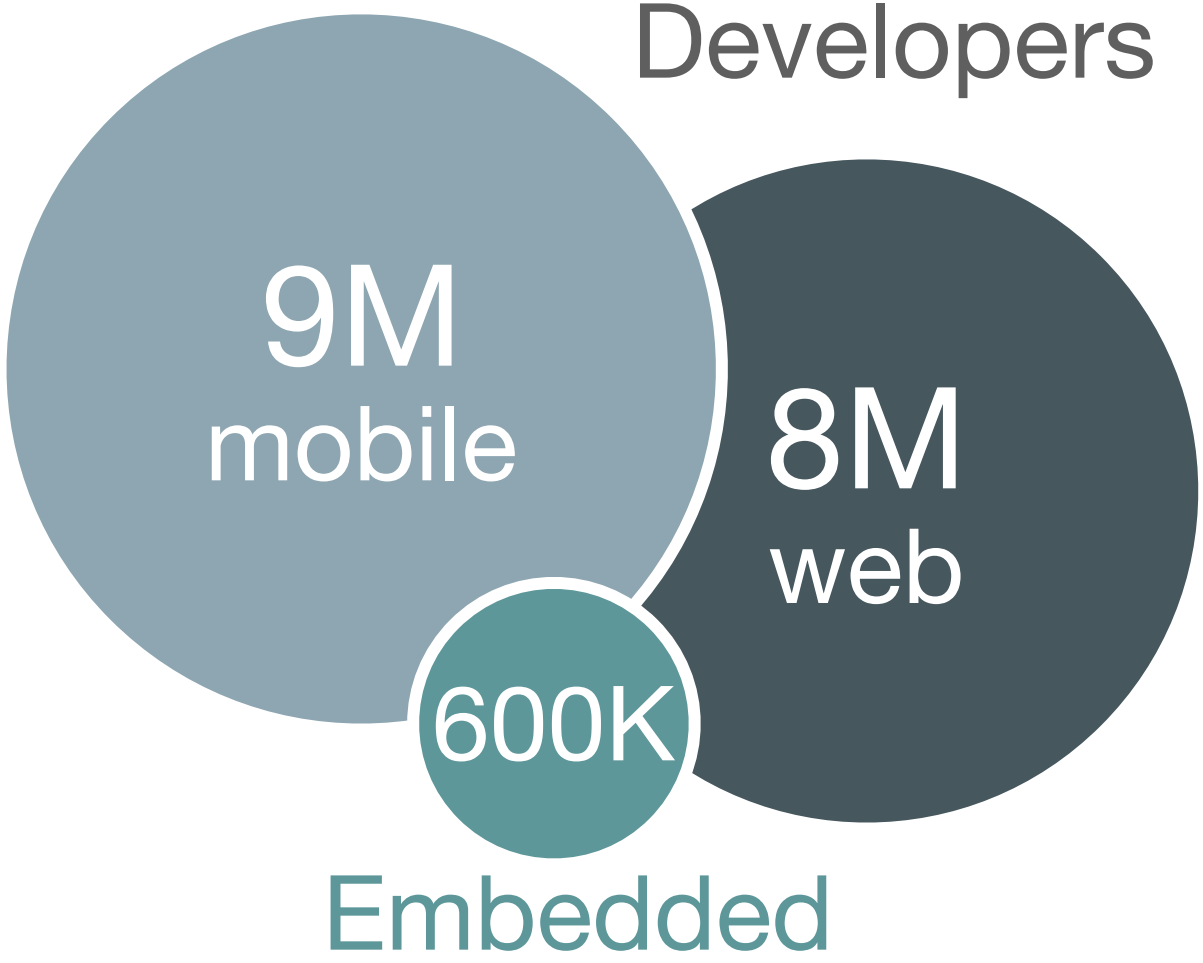
Agenda

- Kona overview
- What is IoT?
- Why does it matter for Java developers?
- CoAP - the web for constrained devices and networks
- Bluetooth

Kona Overview

- OpenJDK project
- Umbrella project for IoT protocols
  - CoAP
  - Bluetooth (coming)
  - Others
- http://openjdk.java.net/projects/kona

IoT will be built by a new class of developer



IoT startup investment in 2015

VC investment & expected investment, Pitchbook.com & ARM estimates, 2015

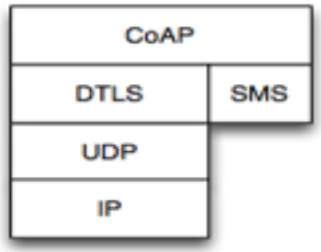
Rapid innovation in IoT

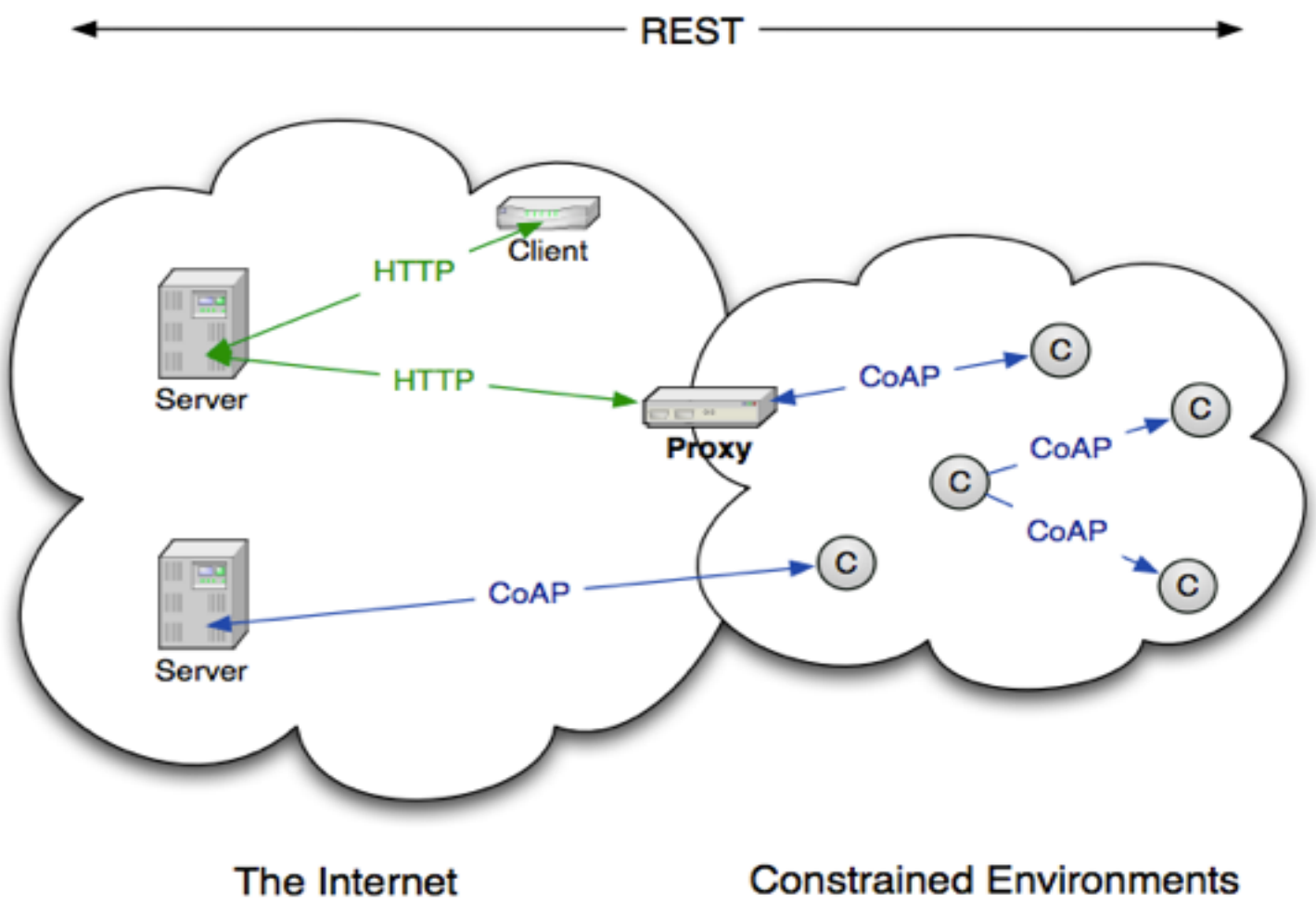
www.kickstarter.com/ARM

CoAP (Constrained Application Protocol)

CoAP: The Web of Things Protocol

- Open IETF Standard (RFC7252)
- Compact 4-byte Header
- UDP, SMS, (TCP) Support
- Strong DTLS Security
- Asynchronous Subscription
- Built-in Discovery

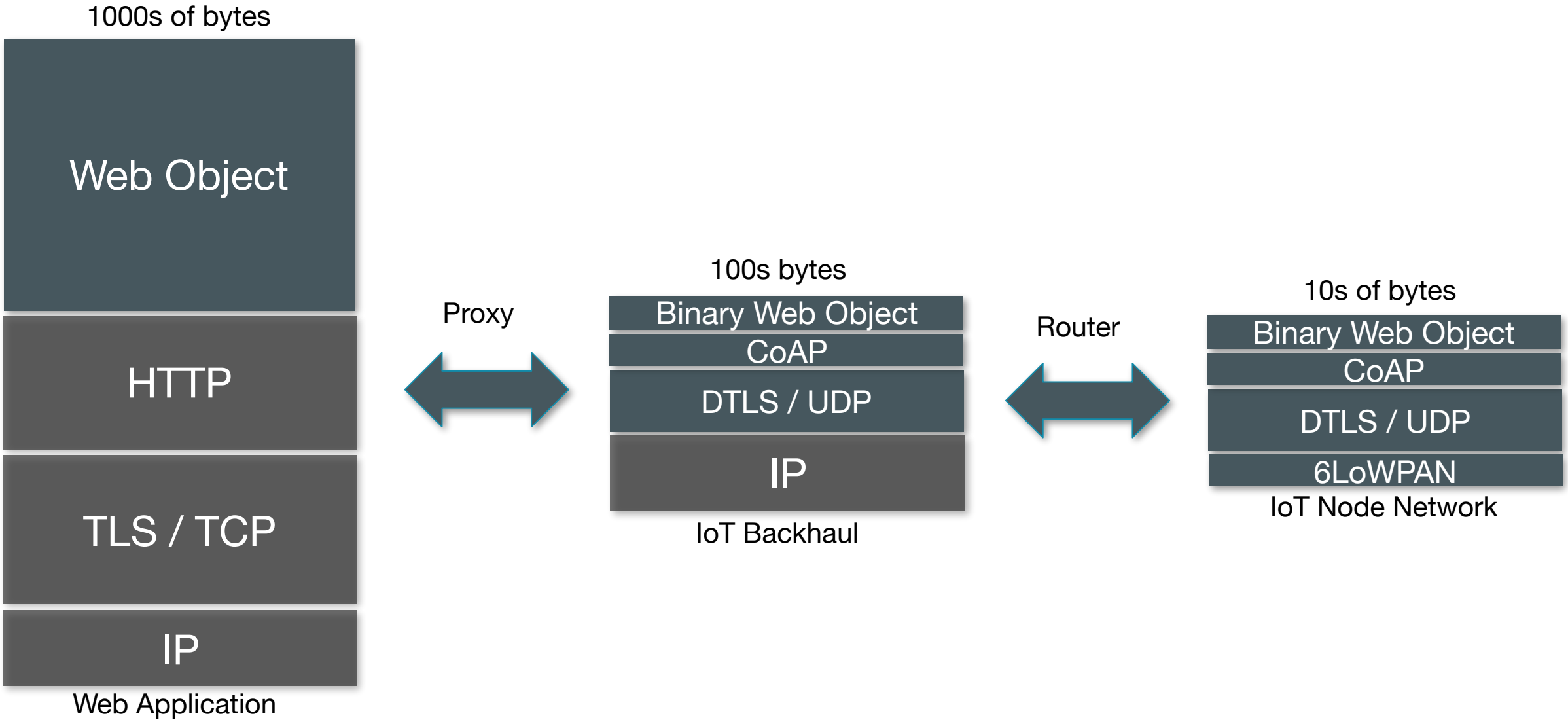




IETF

JavaOne ORACLE

From Web Applications to IoT Nodes



The Kona CoAP Library

- The world longest running CoAP implementation, since 2010
- Developed at Sensinode and then ARM, widespread production use
- Location: http://hg.openjdk.java.net/kona/coap
- Implements:
  - RFC7252 (CoAP)
  - RFC6690 (LinkFormat)
  - draft-ietf-core-observe
  - draft-ietf-core-block

Creating CoAP a server

```
//create CoAP server with builder
CoapServer server = CoapServer.builder().transport(5683).build();

//add request handler
server.addRequestHandler(”/temp", new SimpleCoapResource("70 F"));
server.addRequestHandler("/pwr", new SimpleCoapResource("2 W"));

//start server
server.start();
```

Hello world CoAP resource

```
class HelloWorldCoapResource extends CoapResource {
    private String body = "Hello World";

    @Override
    public void get(CoapExchange ex) throws CoapCodeException {
        ex.setResponseBody(body);
        ex.setResponseCode(Code.C205_CONTENT);
        ex.sendResponse();
    }
    @Override
    public void put(CoapExchange ex) throws CoapCodeException {
        body = ex.getRequestBodyString();
        ex.setResponseCode(Code.C204_CHANGED);
        ex.sendResponse();
    }
}
```

Creating a CoAP client

```
//create client with a builder
CoapClient client = CoapClientBuilder.newBuilder(
      new InetSocketAddress(”localhost", 5683)).build();

//GET request
CoapPacket coapResp = client.resource(”/temp").get().get();

//PUT request
coapResp = client.resource("/a/relay")
      .payload("1", MediaTypes.CT_TEXT_PLAIN).put().get();

//non blocking request
client.resource("/a/relay").payload("1", MediaTypes.CT_TEXT_PLAIN).get()
                .thenAcceptAsync(System.out::println);

//it is important to close connection in order to release socket
client.close();
```

Bluetooth

Bluetooth Agenda

- OpenJDK Project
- Overview
- API Review
- Class Review

OpenJDK Kona Bluetooth Project

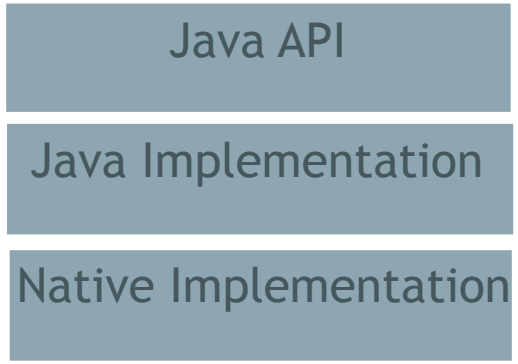
- API/Java Implementation/Native Implementation source code
- Build using make
- Unit tests
- Functional tests
- Samples
- README
- Javadoc

Bluetooth Overview

- New project
- Goal is an easy, modern API for working with Bluetooth devices
- Supported platforms
  - Linux/x86
  - Linux/ARM
- Requires BlueZ
  - > 4.98 & < 4.101

Bluetooth Overview (cont'd.)

- Uses latest JDK8 language features (e.g. lambda's)
- Bluetooth Classic and LE
- Native Code
  - Native API layer for portability
  - Interfaces with BlueZ



Bluetooth API Packages

- Packages
  - jdk.bluetooth - core API
  - jdk.bluetooth.serial - Bluetooth Serial API
    - Bluetooth Serial Port Profile (SPP v1.1)

Bluetooth API Core Classes

- LocalDevice
  - Interface to the local Bluetooth adapter
  - Get paired/discovered devices
  - Set the local adapter to be pairable/discoverable
- RemoteDevice
  - Represents a Bluetooth device
  - Pair/un-pair
  - Search for services
  - Get information about the device

Bluetooth API Core Classes

- BluetoothPermission
  - Permissions for discovering, pairing, power, and searching
- BluetoothChannel
  - A communications connection between two devices.
  - Get Input/OutputStream to send/receive data.

Discovery Example

```
LocalDevice localDevice = LocalDevice.getDefault();

// MyDiscoveryConsumer
private Consumer<RemoteDevice> discoveryConsumer = new Consumer<RemoteDevice>() {
    public void accept(RemoteDevice remoteDevice) {
        System.out.println("deviceDiscovered: " + remoteDevice.getName());
    }
};


// MyDiscoveryApp
try {
    discoveryFuture = defaultLocalDevice.discover(discoveryConsumer);
} catch (BluetoothException ex) {
    ex.printStackTrace();
}
```

Pairing Example

```
private byte[] devicePin = “123".getBytes("UTF-8");

// MyPairingHandler
private Function<Void,byte[]> pairPinRequestHandler = new Function<Void,byte[]>() {
    public byte[] apply(Void v) {
        System.out.print("Returning PIN: " + devicePin);
        return devicePin;
    }
};


// MyPairingApp
try {
    pairFuture = remoteDevice.pair(pairPinRequestHandler);
    System.out.println("Pairing has started.");
} catch (BluetoothStateException bse) {
    bse.printStackTrace();
}
```

Kona

- Contribute! Open-Source
- Get the code: git co http://hg.openjdk.java.net/kona
- Where to find more information
  - http://openjdk.java.net/projects/kona

- Q & A