

Féidearthachtaí as Cuimse
Infinite Possibilities

Week 11

Advanced architectures

Fundamentals of IoT
Dr. Eoin Rogers (eoin.rogers@tudublin.ie)



Lesson Outline

- Now that we've seen more complex IoT systems, it's useful to think about what the architecture of such systems might look like!
- We'll examine the architectures of some real IoT systems
- We'll also look at protocol buffers, a form of data encoding developed by Google

Architectures

Can anyone name the three architectural styles?

Architectural styles

- Device-to-cloud
- Device-to-gateway
- Device-to-device

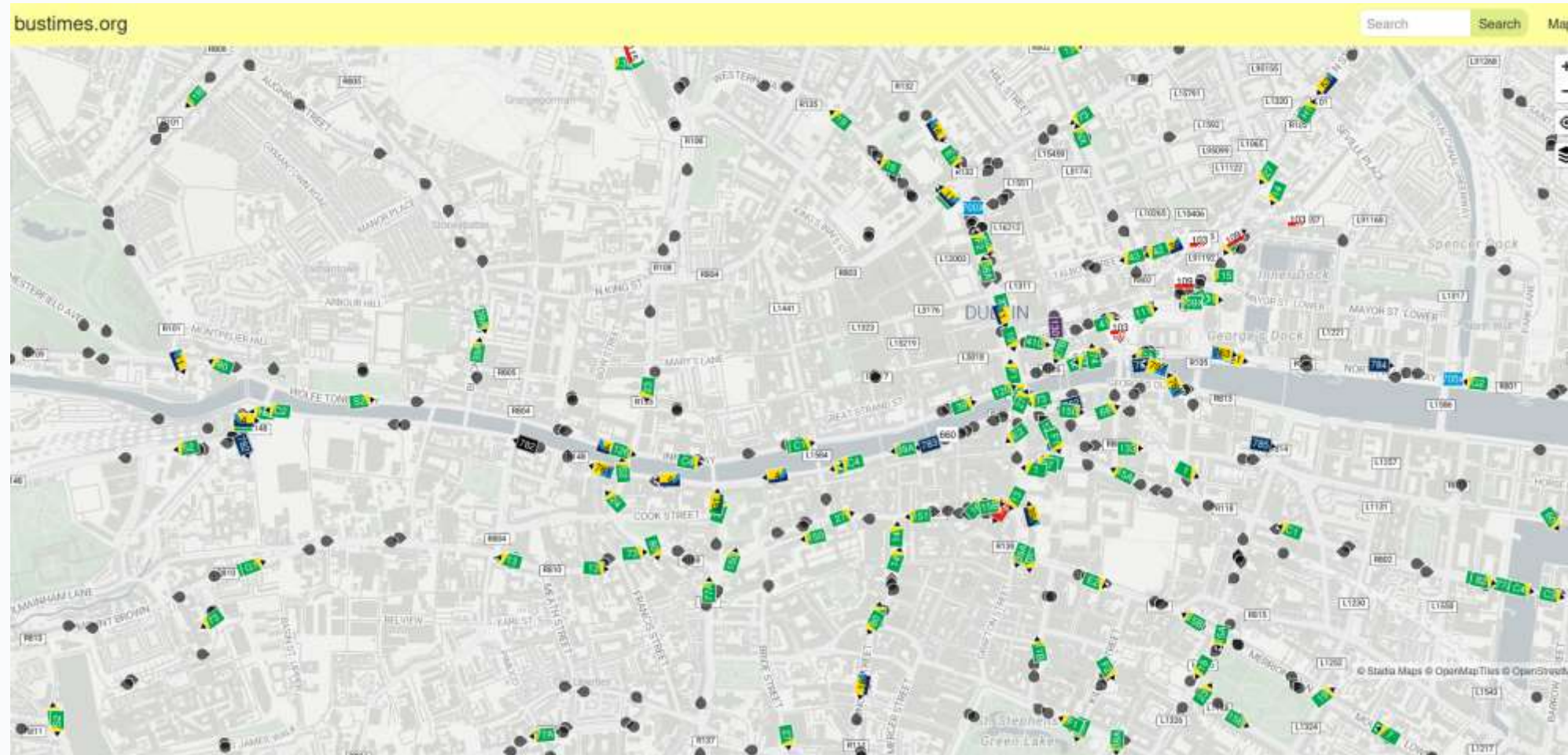
Three tier architecture of IoT devices

- **Tier 1** – Edge devices
- **Tier 2** – Gateway devices
- **Tier 3** – Cloud systems

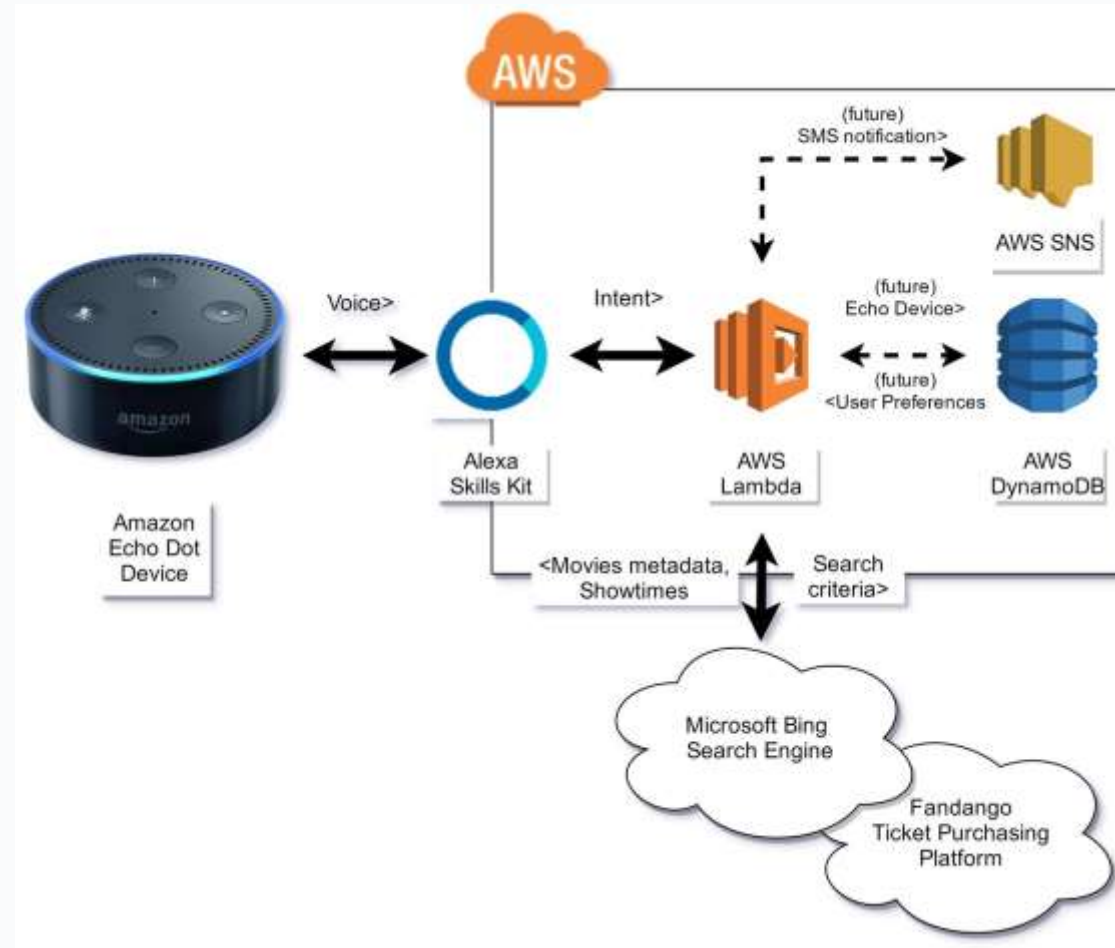
Case studies

Three examples of IoT systems

Bus Times



Amazon Echo/Alexa



Alexa Skills Kit (ASK)

- This is what you use to develop new Alexa **skills**
- Users can add them to an Echo they own via an **online store**
- The skills then run on an **AWS Lambda** service

Recognising speech commands

- Internally, the system uses a **neural net** to convert the speech into text, which is matched against **sample utterances** that should trigger the skill
- Sample utterances can contain **slots** (variables you want to show up in the intent containing a word or phrase of interest)
- These can then be converted to data structures called **intents**

The AbstractRequestHandler class

- You can write code in Python or JS to respond to an intent
- The code should derive from a class called AbstractRequestHandler, and you should override two methods to implement the functionality:
 - **can_handle()** -- Returns true iff your code can handle this intent
 - **handle()** -- Which will actually compute the response

Our IoT temperature system

- What would the architecture look like for this system?
- How would you document it in a diagram?

Protocol buffers

An efficient system for sending data over the internet

Protocol buffers overview

- Developed by Google in 2001, open sourced in 2008
- Allows you to define a **schema** and encode/decode messages in it into a **binary data format** that is small and low bandwidth
- A protocol buffer compiler (**protoc**) auto-generates the code for this for you: there is **no need to write low-level bit manipulation code** yourself...

Protocol buffer schemas

```
edition = "2023";

package tutorial;

message Person {
  string name = 1;
  int32 id = 2;
  string email = 3;

  enum PhoneType {
    PHONE_TYPE_UNSPECIFIED = 0;
    PHONE_TYPE_MOBILE = 1;
    PHONE_TYPE_HOME = 2;
    PHONE_TYPE_WORK = 3;
  }
}
```

```
message PhoneNumber {
  string number = 1;
  PhoneType type = 2;
}

repeated PhoneNumber phones = 4;

message AddressBook {
  repeated Person people = 1;
}
```

Using the generated code in Python

```
import addressbook_pb2
person = addressbook_pb2.Person()
person.id = 1234
person.name = "John Doe"
person.email = "jdoe@example.com"
phone = person.phones.add()
phone.number = "555-4321"
phone.type = addressbook_pb2.Person.PHONE_TYPE_HOME
```

gRPC

- Protobufs are also the basis of Google's Remote Procedure Call protocol (**gRPC**), which is widely used in industry
- The normal protobuf language allows you to specify procedures in the form of **function signatures**
- This is used to generate both client and server-side code. The **server code is then modified to implement the desired behaviour**

gRPC example

```
// The greeting service definition.
service Greeter {
    // Sends a greeting
    rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
    string name = 1;
}

// The response message containing the greetings
message HelloReply {
    string message = 1;
}
```

Summary

- Reviewed the architectural styles common IoT systems
- Introduced the idea of three tiered systems, which is a common way of thinking about IoT systems
- Discussed the architecture of three IoT systems, and described how you might document and communicate this information
- Provided an introduction to Google's protobuf system

That's all for this week

Thanks for your attention!