Microservices and IoT

David Bosschaert and Carsten Ziegeler

I am doing IoT !

D

I am doing IoT !

D

David Bosschaert

- R&D Adobe Ireland
- Co-chair OSGi Enterprise Expert Group
- Apache Felix, Aries PMC member and committer
- … other opensource projects
- Cloud and embedded computing enthusiast

D

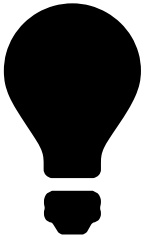
Carsten Ziegeler

- RnD Adobe Research Switzerland
- Team Lead / Founder of Adobe Granite
- Member of the Apache Software Foundation
- VP of Apache Felix and Sling
- OSGi Expert Groups and Board member

C

IoT

IoT Hello World!



C

I am doing IoT ?

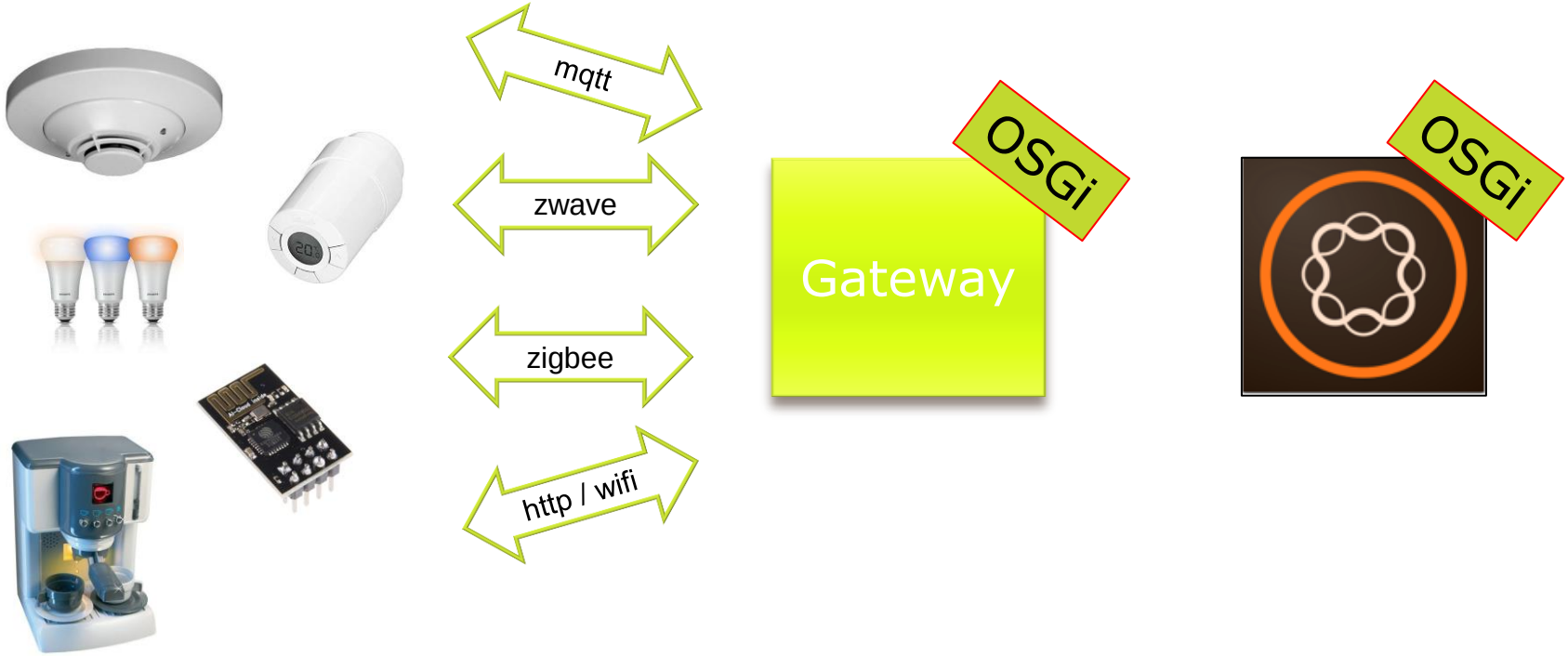
D

IoT and AEM

?

D

IoT and AEM



D

OSGi for Gateway and Cloud

- Unified platform
- Share & Reuse
- Provisioning, tooling...
- Dynamically updatable

C

OSGi Device Abstraction Layer

- Devices and Functions
- Protocol independent

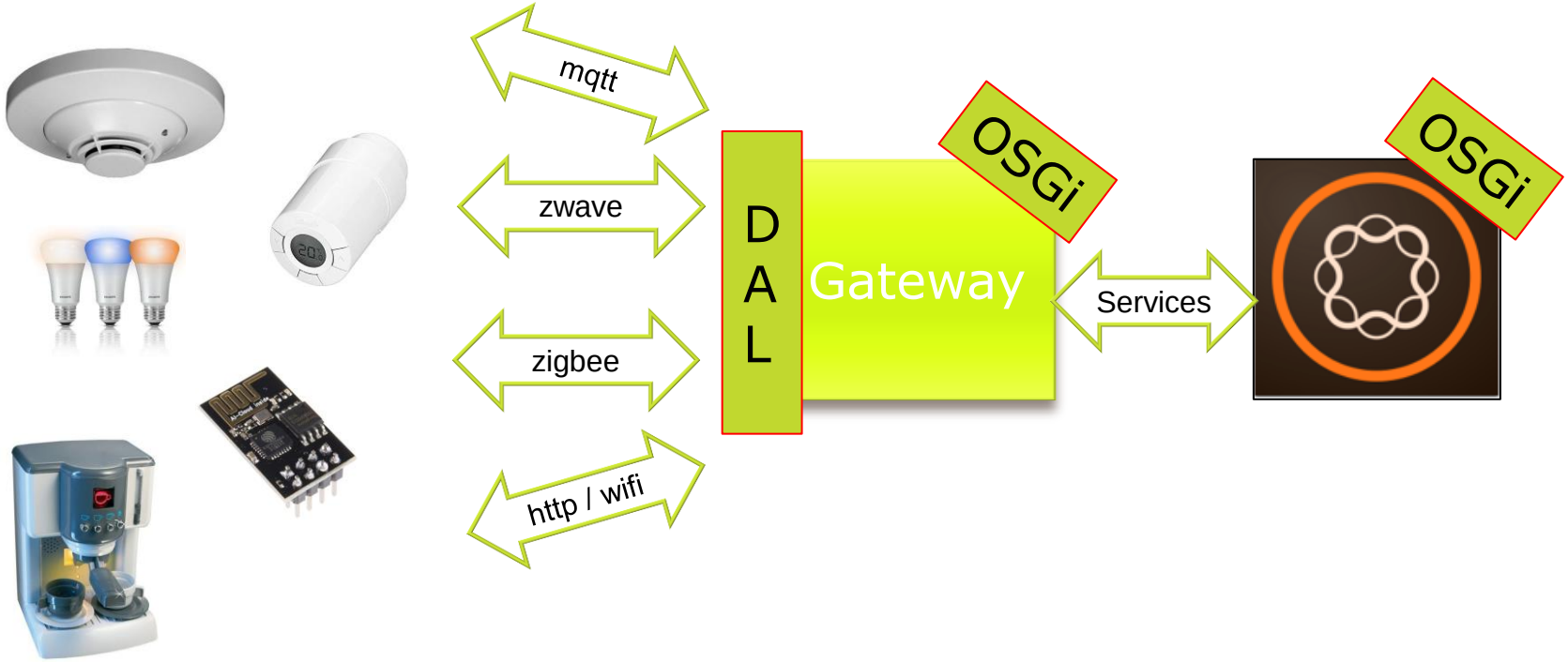
C

OSGi IoT

- Protocols (CoAP, MQTT)
- Zigbee, EnOcean
- Not limited to this

C

IoT



C

Adobe

And now Camera, Lights, and…ACTION

D

Demo Scenario

Photo credit:
www.bargainmoose.ca

D

Adobe

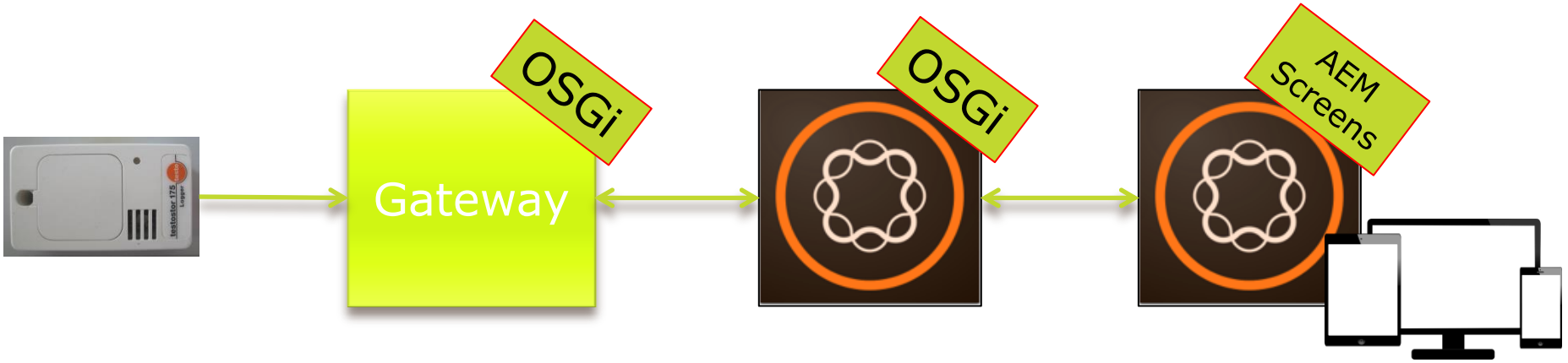
Increase sale discount with the weather!

Clipart credit: pixabay.com

Photo credit: www.bargainmoose.ca

D

Architecture



D

Adobe

Microservices

Microservices

- Do one thing but do it right
- Resource optimizations
- Focused development/testing
- Reduced time to production

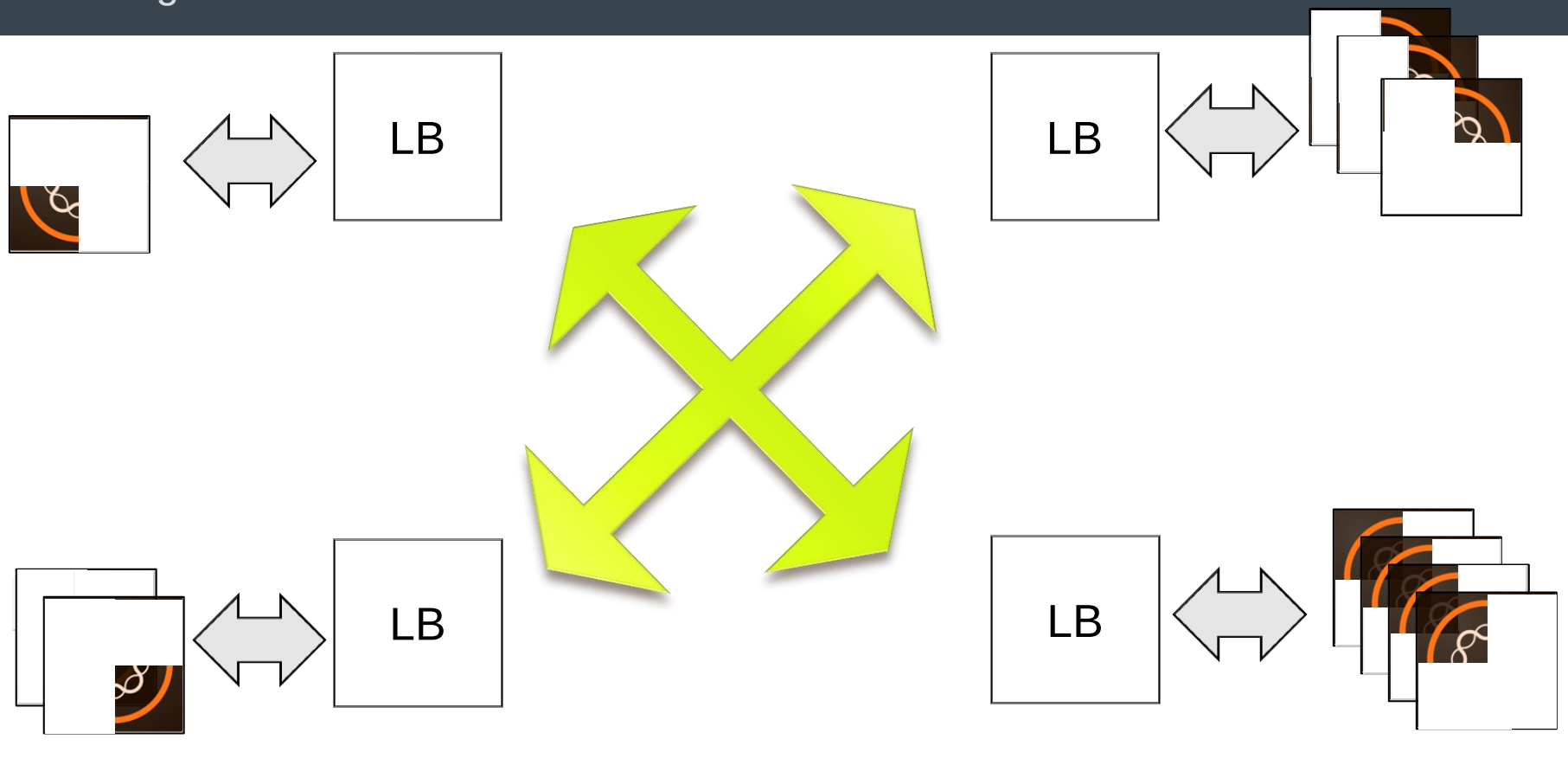
C

Microservices

C

Adobe

Scaling



Orchestration

- Various solutions
- Containers
- The twelve-factor app
- Disposability

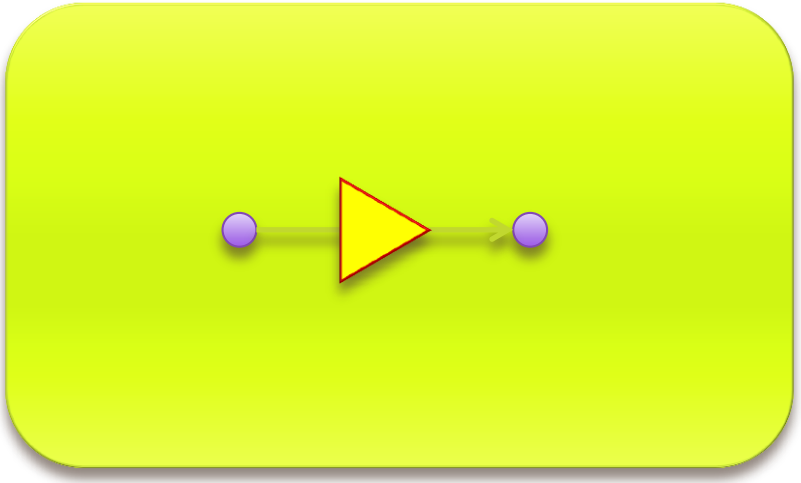
C

Communication

- Discovery / Topology
- REST
- Distributed OSGi

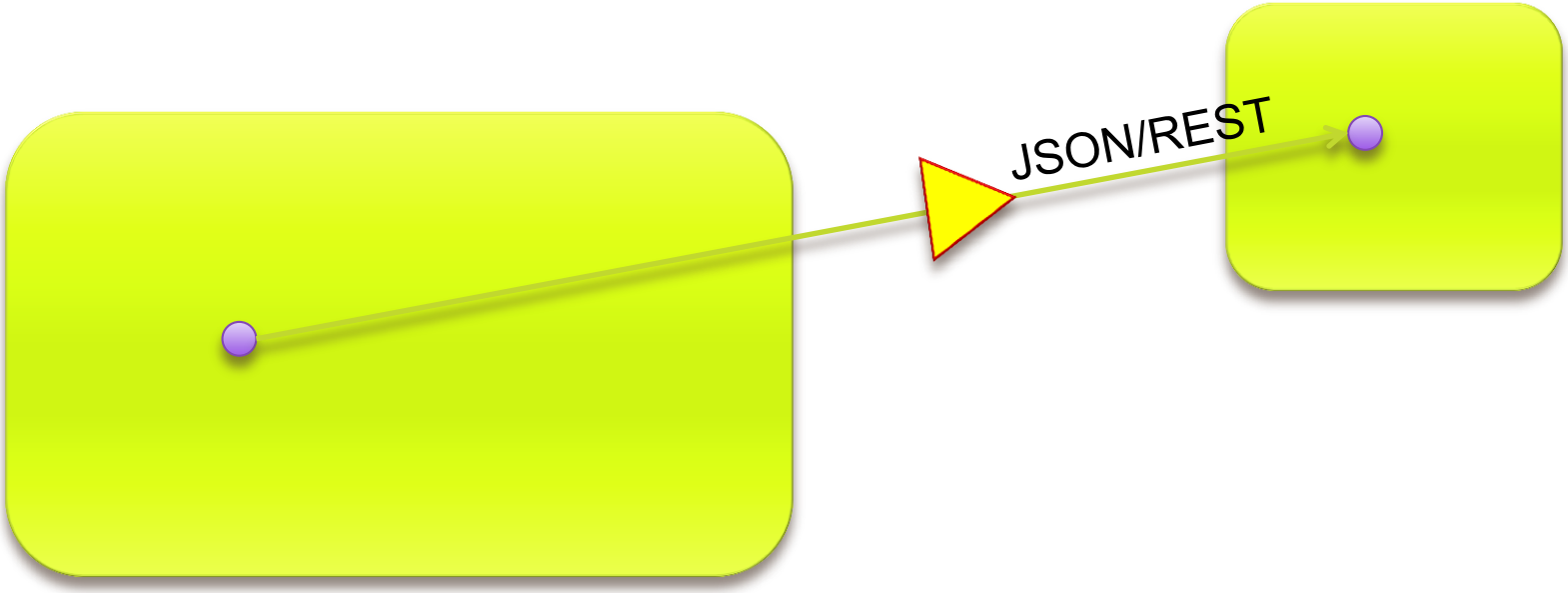
D

OSGi Services



D

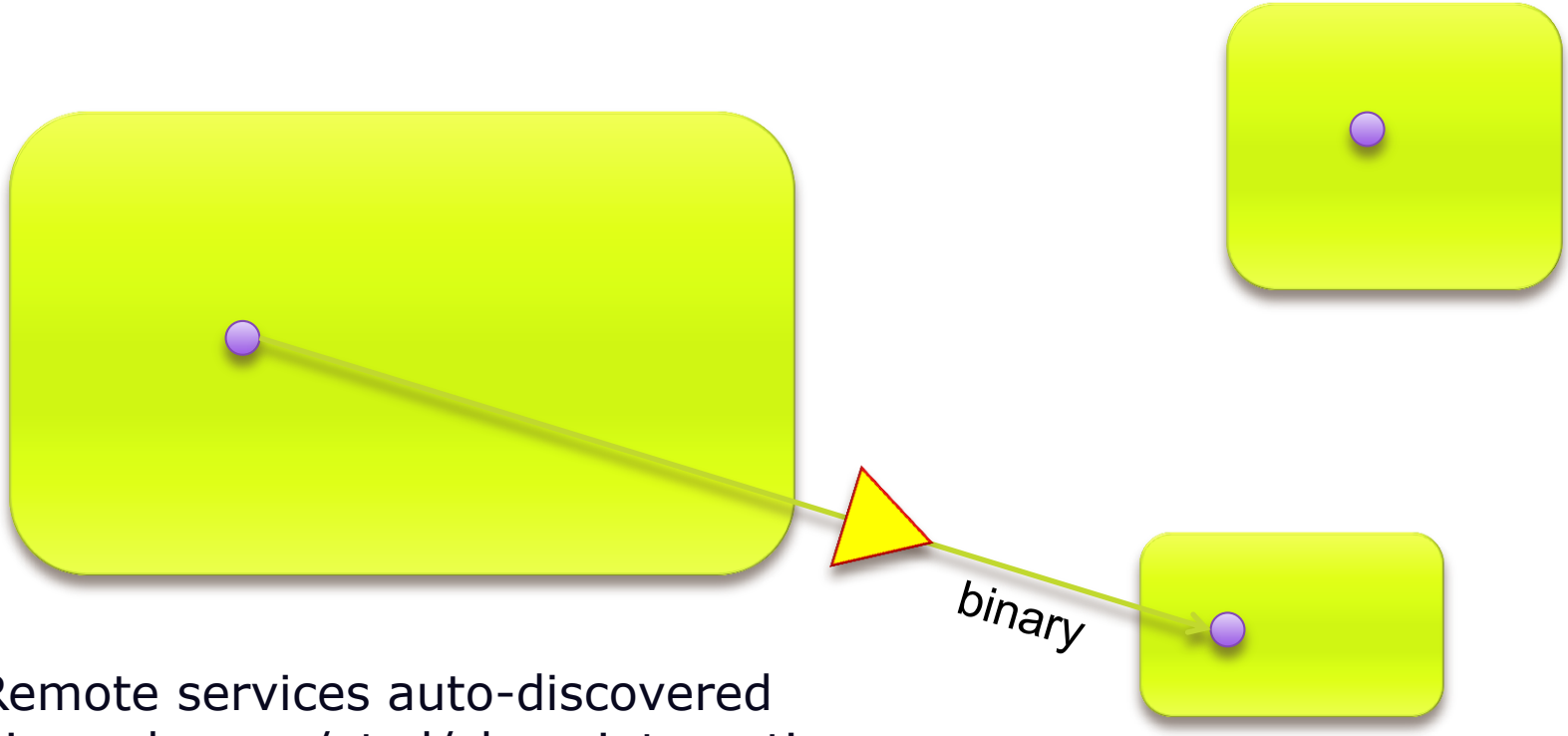
OSGi Remote Services



D

Adobe

Remote Services – dynamic rebinding



Remote services auto-discovered
via zookeeper/etcd/slp … integration

D

Using Local OSGi Services

```
class MyComponent {
  @Reference
  PaymentService paymentService;

  @Activate
  public void activate() {
    paymentService.makePayment(…);
  }
```

C

Using Remote OSGi Services

```
class MyComponent {
  @Reference
  PaymentService paymentService;

  @Activate
  public void activate() {
    paymentService.makePayment(…);
  }
```

C

Registering an OSGi Service

```
@Component(service=PaymentService.class)
class MyPaymentService implements PaymentService {
  public boolean makePayment(…) {
    ...
  }
}
```

C

Registering a Remote OSGi Service

```
@Component(service=PaymentService.class,
  property="service.exported.interfaces=*")
class MyPaymentService implements PaymentService {
  public boolean makePayment(…) {
    ...
  }
}
```

C

Asynchronous Services

```
class MyComponent {
  @Reference
  PaymentService paymentService;

  @Reference
  Async asyncService;

  @Activate
  public void activate() {
    PaymentService mediated =
      asyncService.mediate(paymentService, PaymentService.class);

    asyncService.call(mediated.makePayment(…))
      .then(p -> updatePaymentStatus(p));
    // … thread continues while payment service being called …
```

C

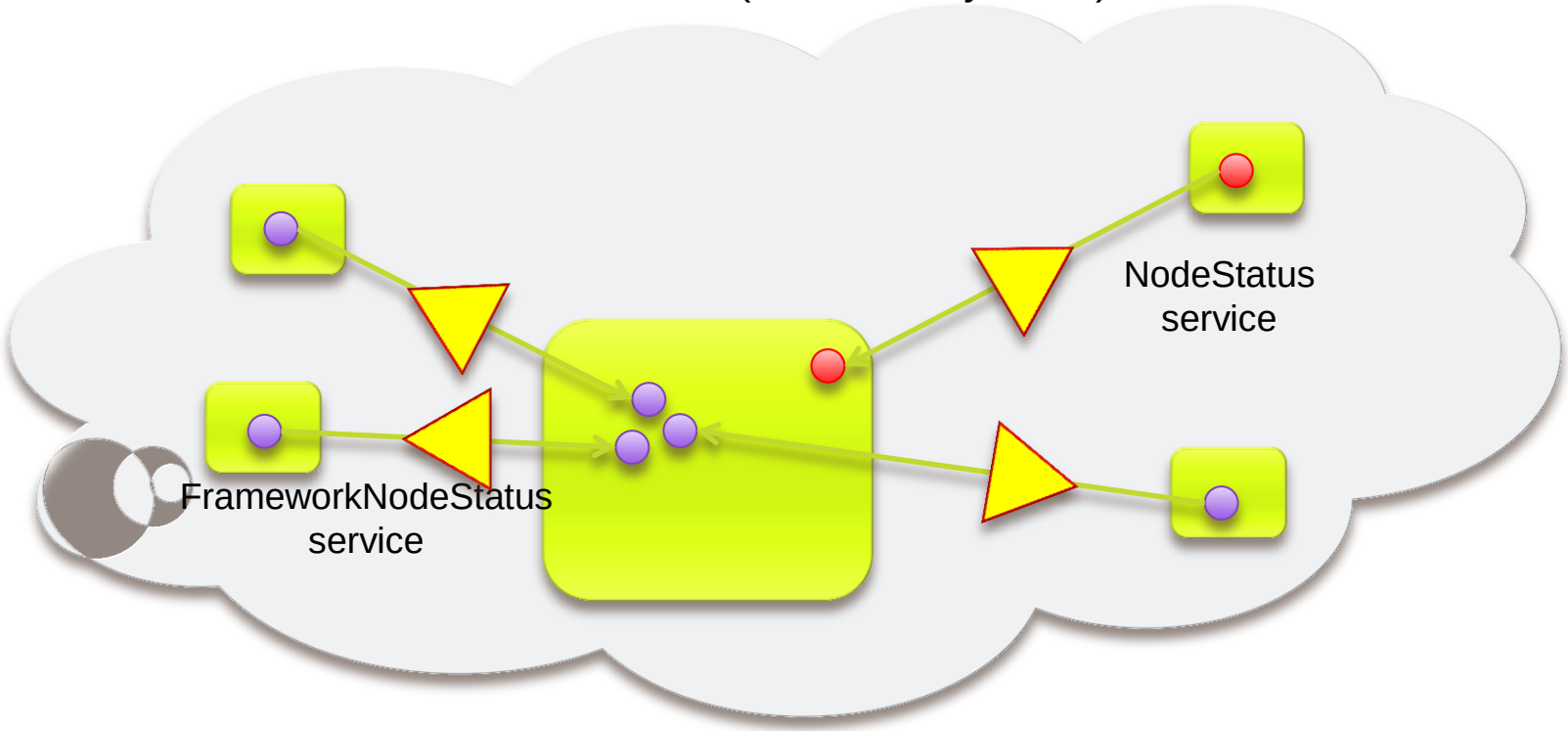
Provisioning

- Cloud : Orchestration of AEM/OSGi
- Gateways : Manage through AEM/OSGi

D

Cloud discovery and provisioning

- RFC 183 Cluster Information (Cloud Ecosystems)



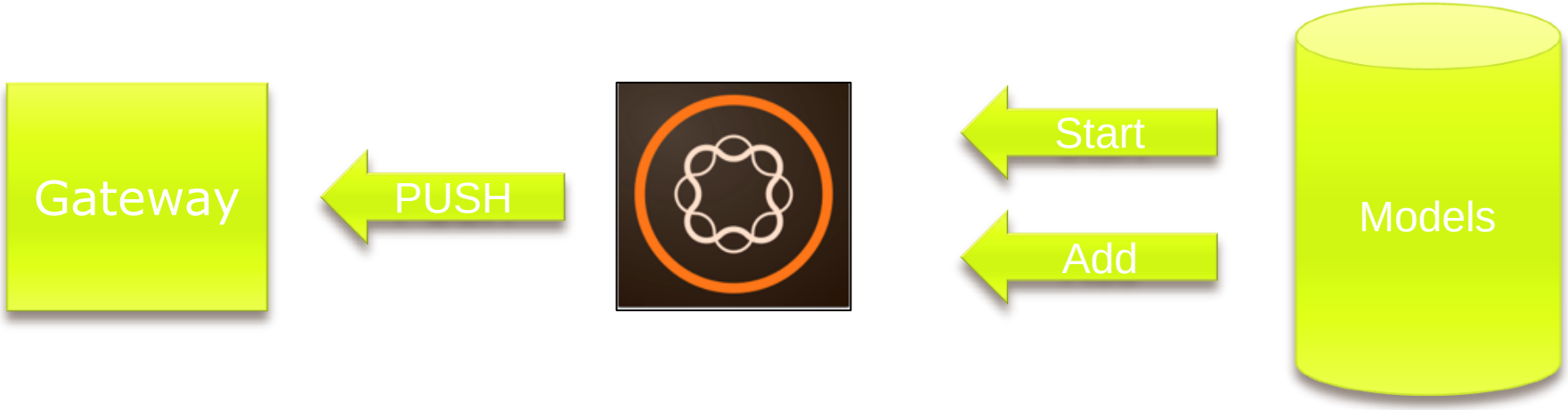
D

Provisioning Model

- Descriptive
- Feature based
- Complete

C

Provisioning



C

Requirements and Capabilities

- Imports / Exports
- Provided / Required Services
- Functionality (whiteboard, extender)
- Extensible (database)

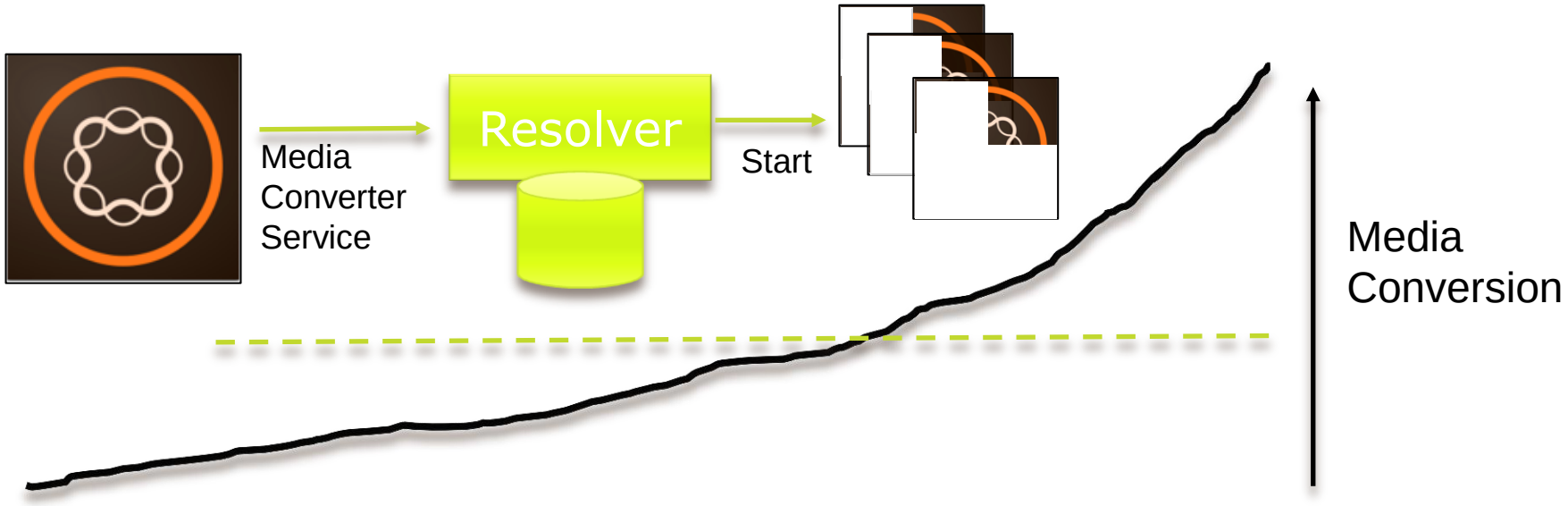
C

OSGi Resolver & Repositories

- Verify runnable configuration
- Calculate required / missing modules

C

Dynamic Provisioning & Orchestration



C

OSGi / AEM is the ideal base for IoT and Microservices

C

THANK YOU.