

Microservices and IoT

David Bosschaert and Carsten Ziegeler



I am doing IoT !

D

I am doing IoT !

D

- 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

- 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

IoT



I am doing IoT ?

D

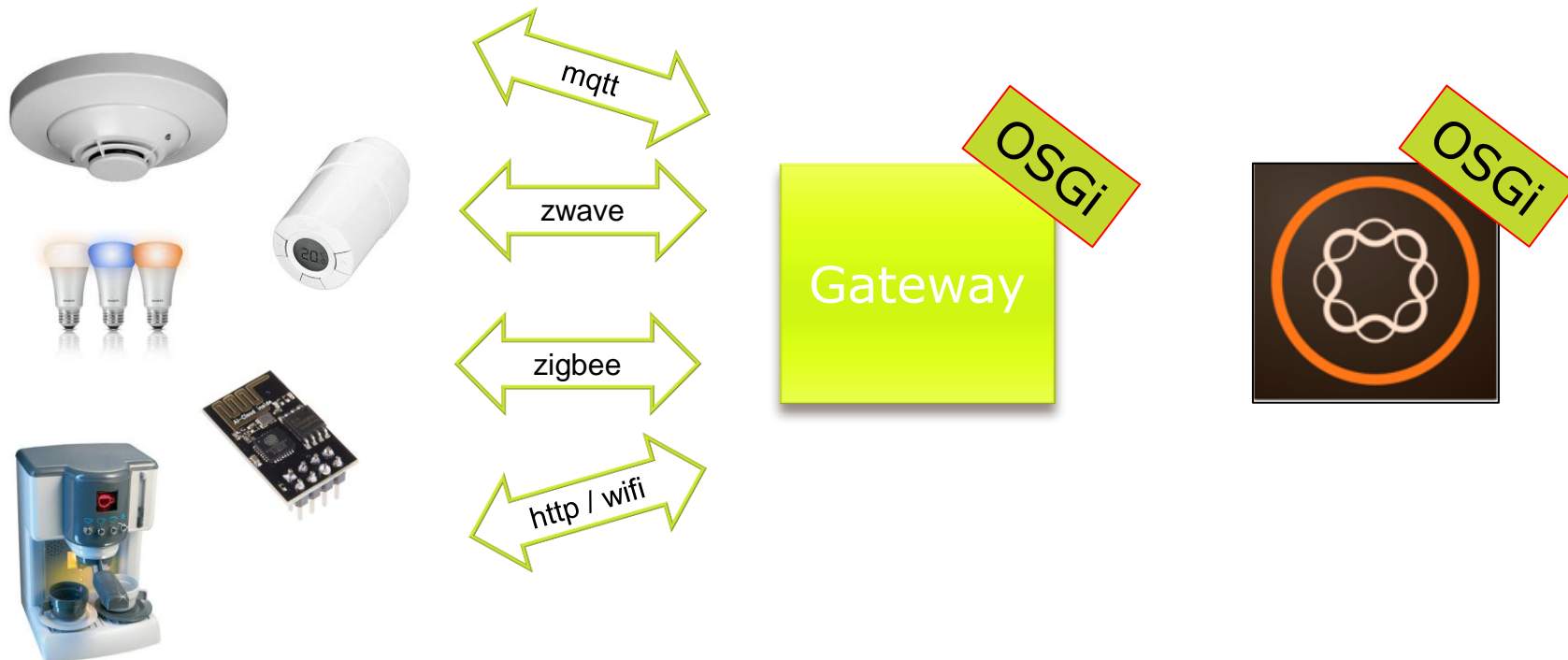


?



D

IoT and AEM

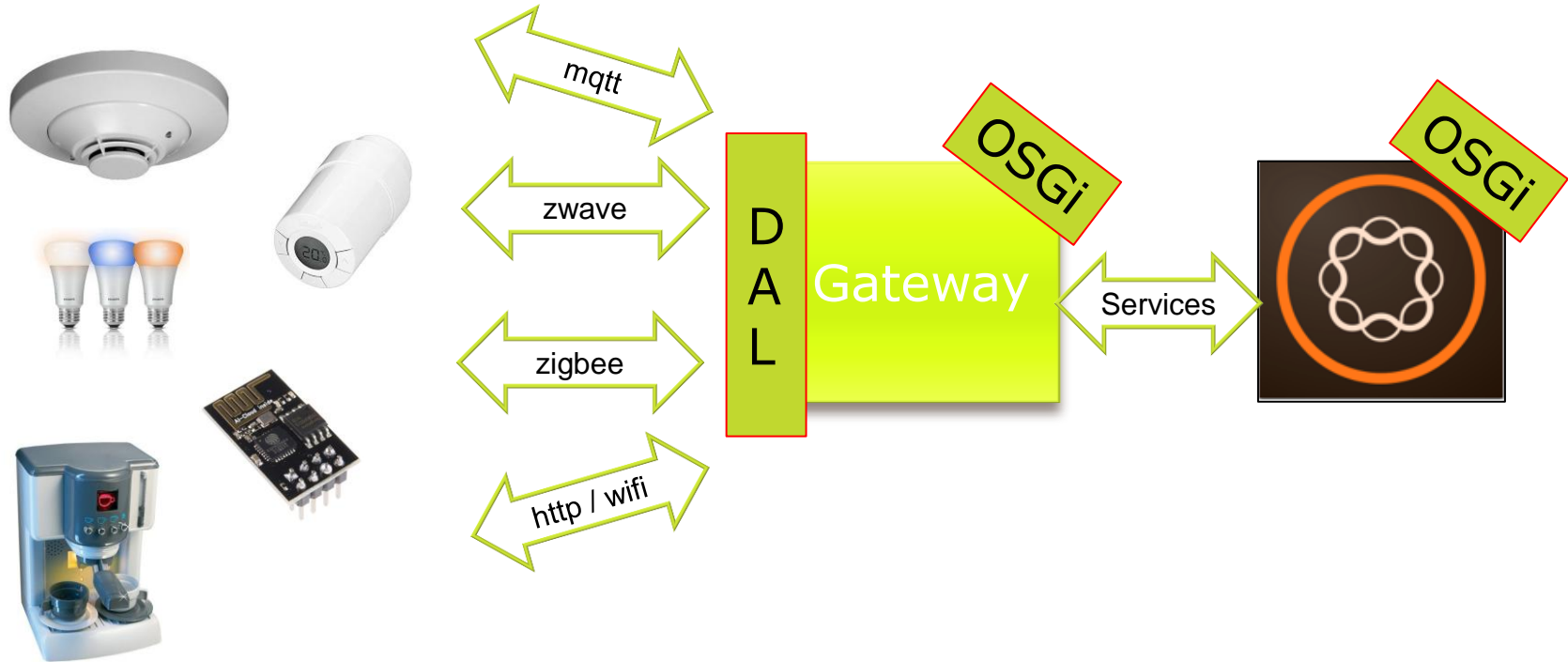


D

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

- Devices and Functions
- Protocol independent

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



*And now
Camera, Lights,
and...ACTION*

D

Demo Scenario



Photo credit:
www.bargainmoose.ca

D

Increase sale discount with the weather!

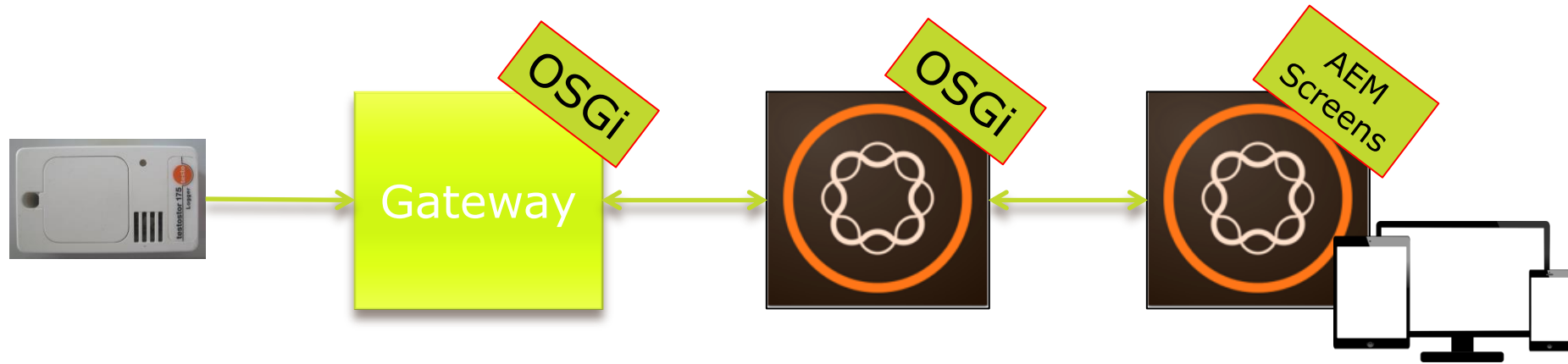


Clipart credit:
pixabay.com

Photo credit:
www.bargainmoose.ca

D

Architecture



D

Microservices

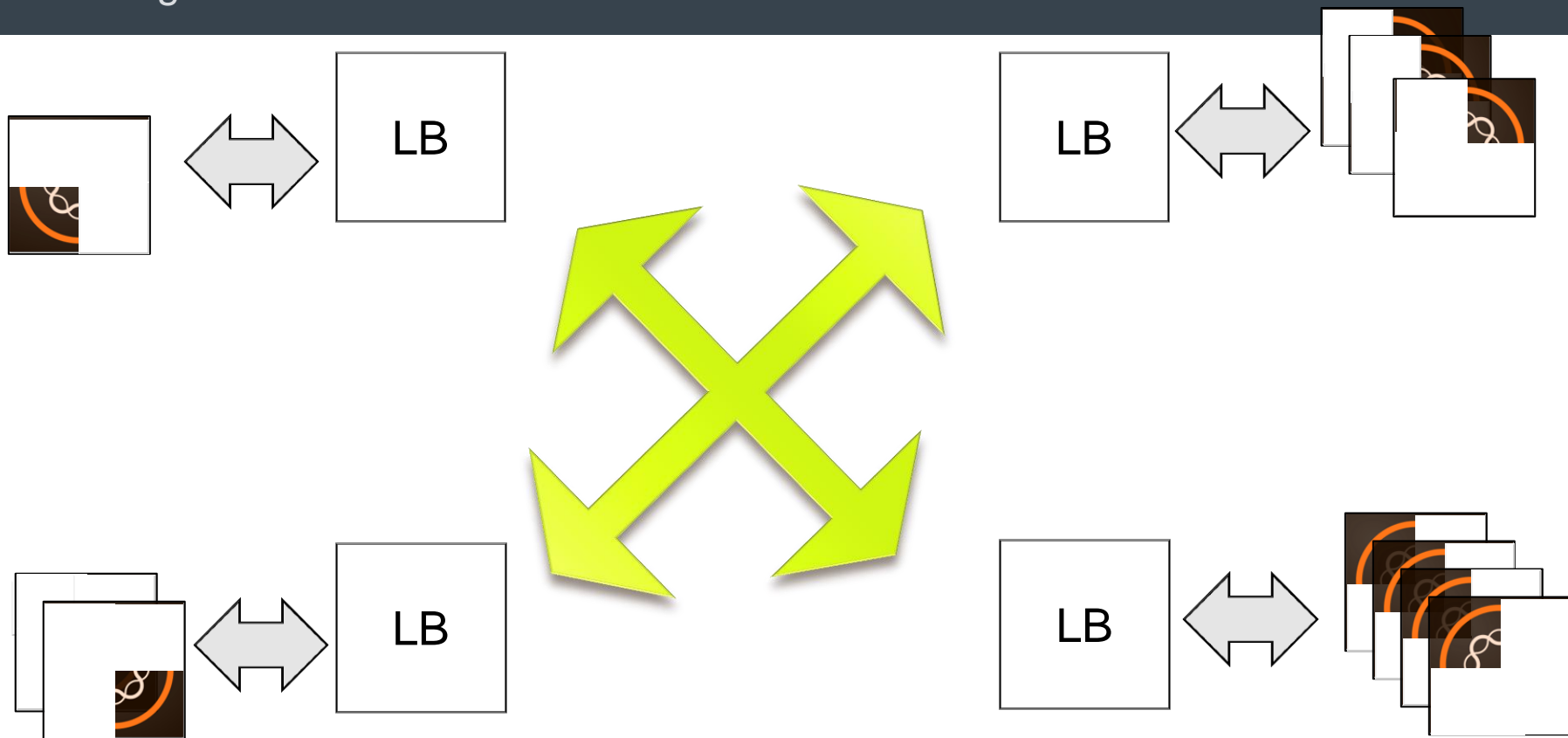
- Do one thing but do it right
- Resource optimizations
- Focused development/testing
- Reduced time to production

Microservices



C

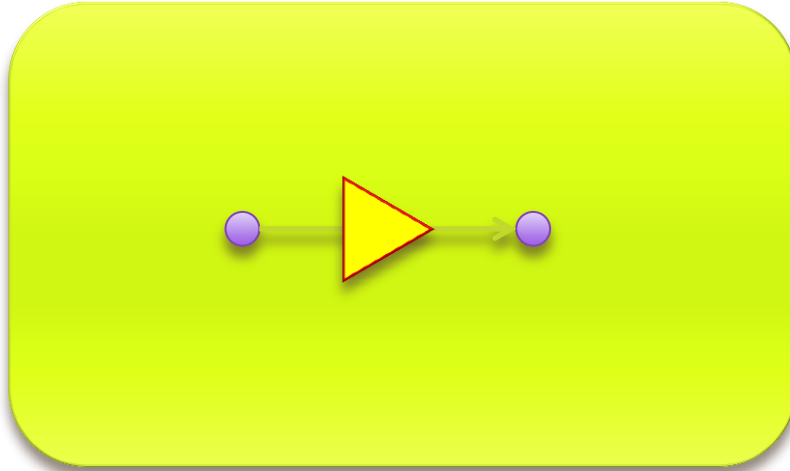
Scaling



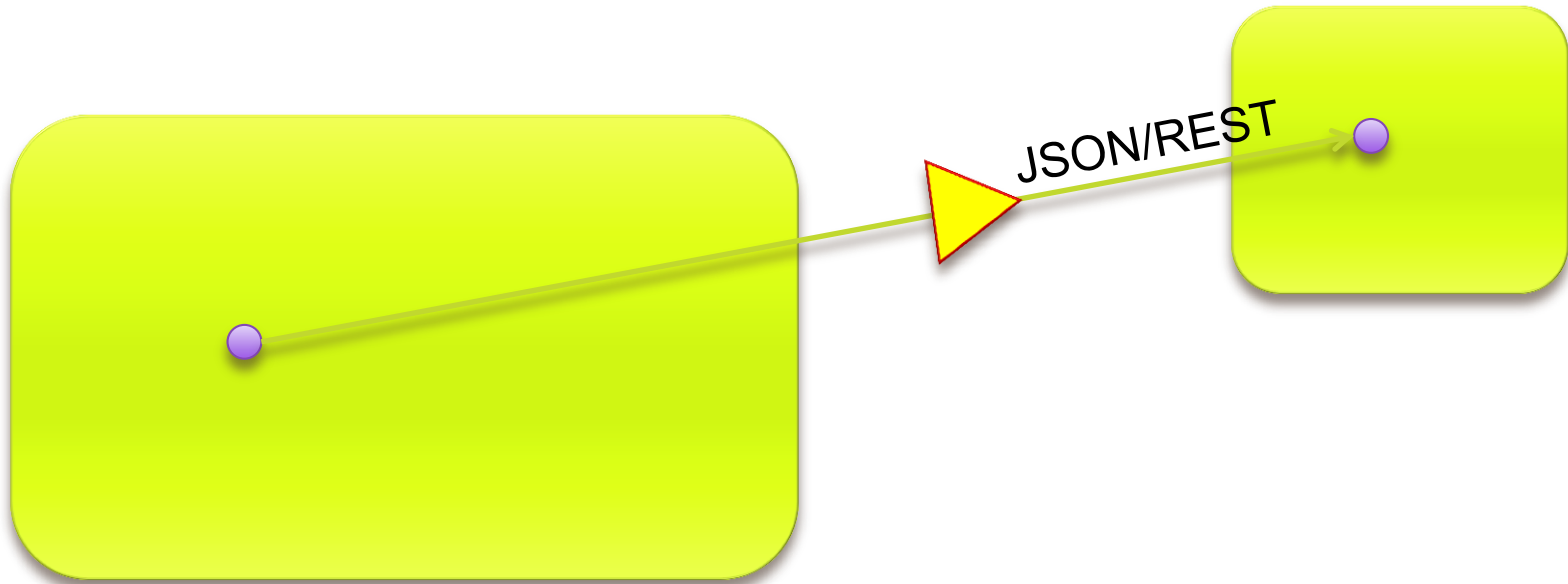
C

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

- Discovery / Topology
- REST
- Distributed OSGi

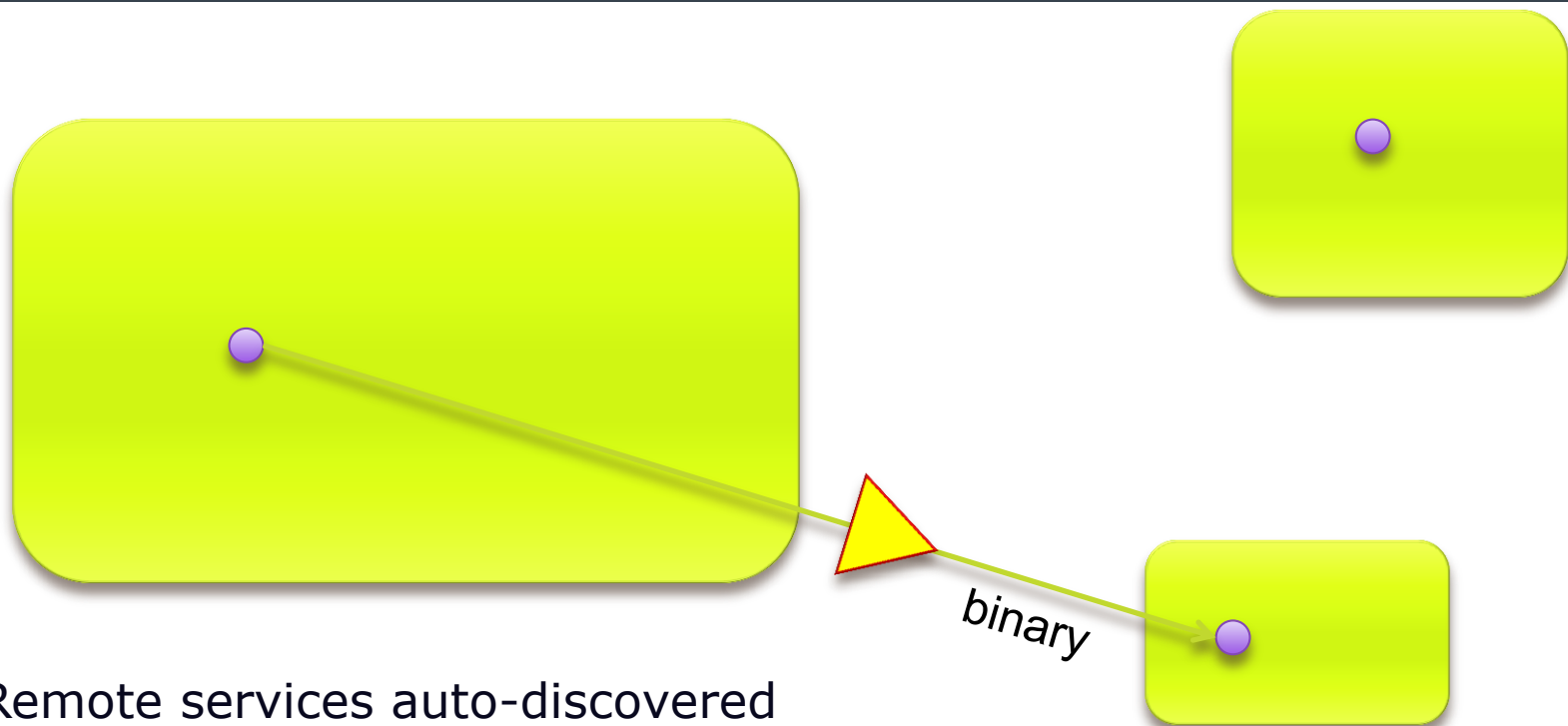


OSGi Remote Services



D

Remote Services – dynamic rebinding



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

D

```
class MyComponent {  
    @Reference  
    PaymentService paymentService;  
  
    @Activate  
    public void activate() {  
        paymentService.makePayment(...);  
    }  
}
```

```
class MyComponent {  
    @Reference  
    PaymentService paymentService;  
  
    @Activate  
    public void activate() {  
        paymentService.makePayment(...);  
    }  
}
```

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

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

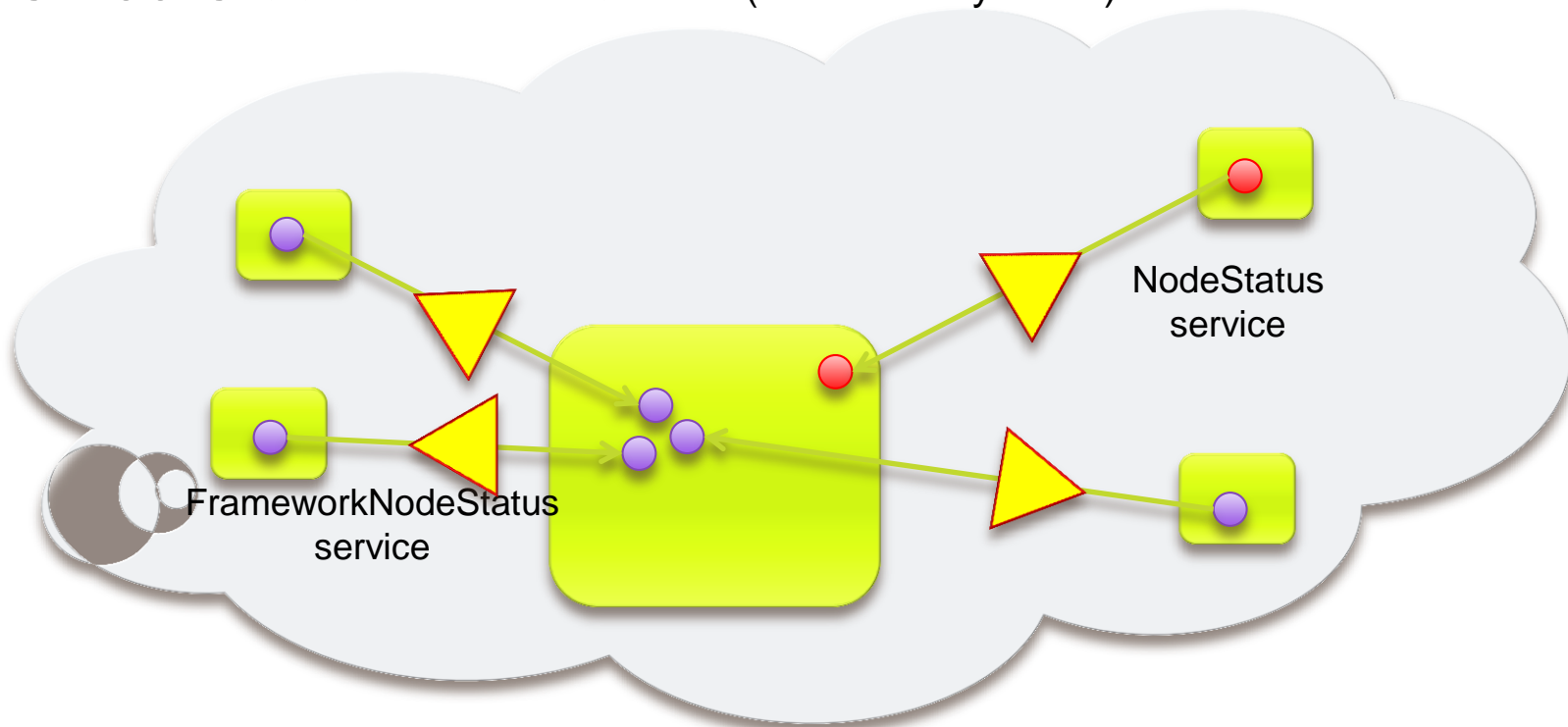
Asynchronous Services

```
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

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

- RFC 183 Cluster Information (Cloud Ecosystems)

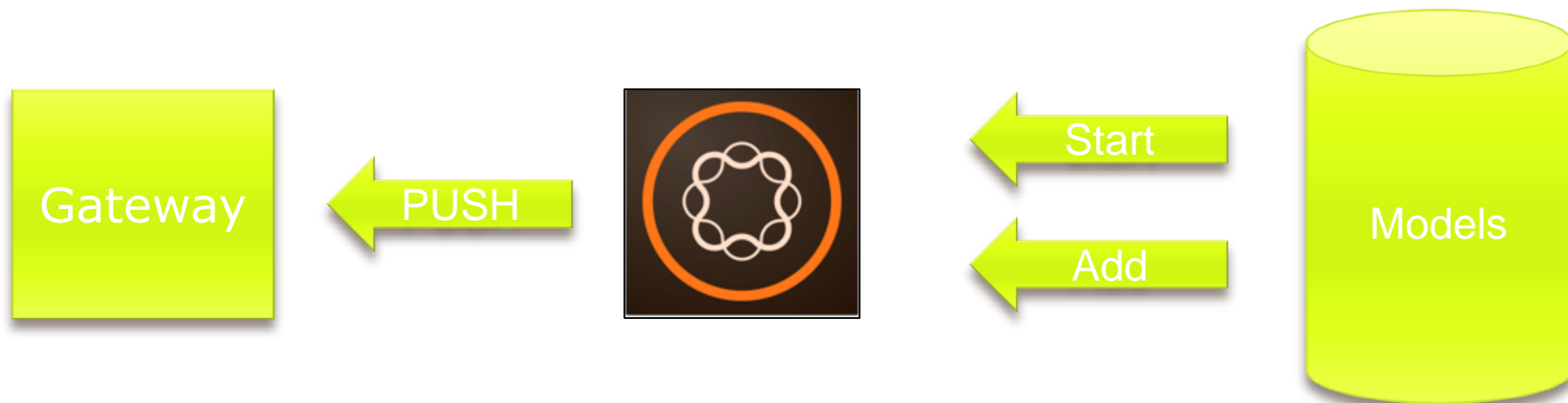


D

Provisioning Model

- Descriptive
- Feature based
- Complete

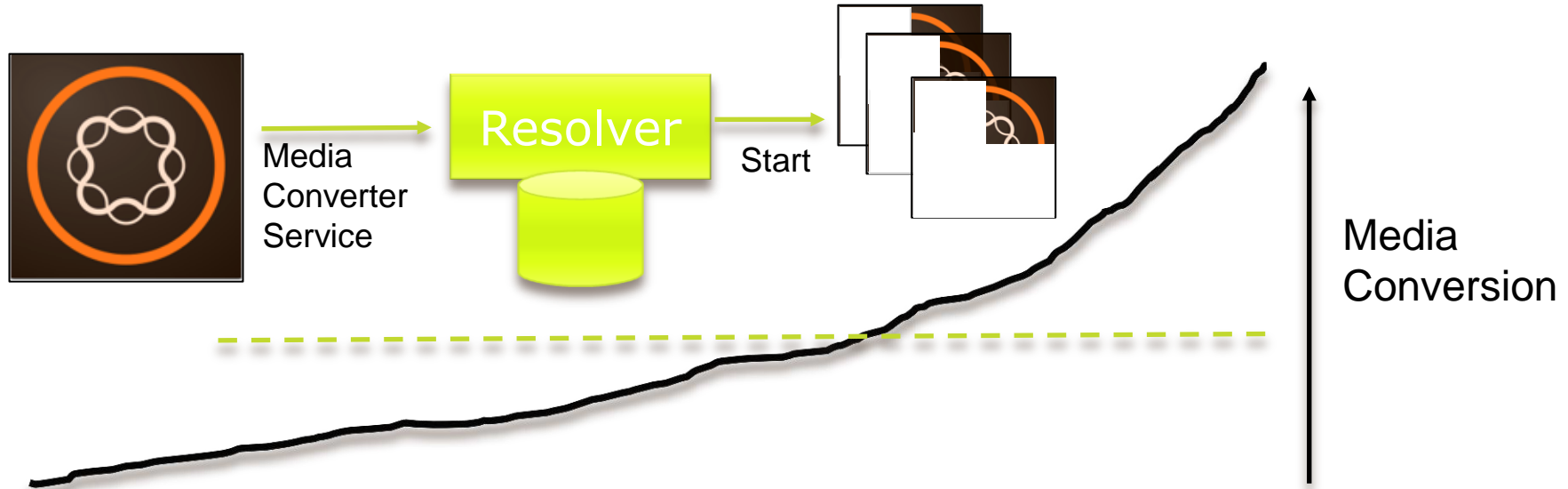
C



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

- Verify runnable configuration
- Calculate required / missing modules

Dynamic Provisioning & Orchestration



C

THANK YOU.