# MicroServices, IoT, and Agile… Seriously!

Fred George
Fred George Consulting
fredgeorge@acm.org
@fgeorge52

APRIL 19-21 | BOSTON, USA

AATC2017

# Fred George

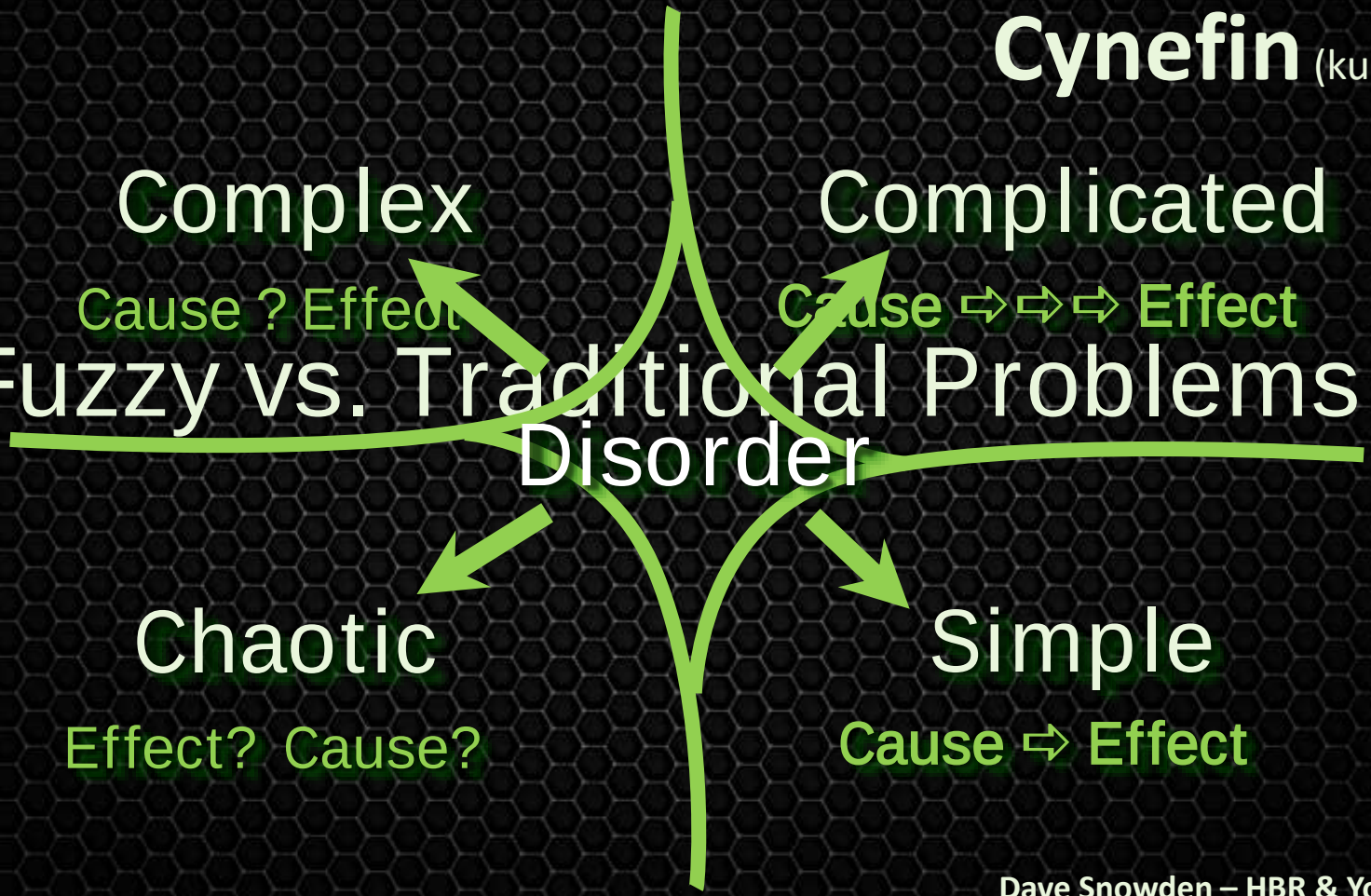
- Programmer
  - Since 1968 (Basic)
  - 65,000 hours experience
  - 70+ languages
  - Computer Science Degree 1973
- Manager
  - 17 years IBM
  - Business degree, MIT Sloan School 1986
  - Product Owner, IBM
  - VP, ThoughtWorks
  - Co-founder, Outpace (Silicon Valley)
  - Senior Advisor to 3 tech companies
- Technologist
  - Computer networks – 70's
  - Token Ring LAN – 80's
  - GUI's – late 80's
  - OO – late 80's
  - Agile – late 90's
  - MicroServices – mid-2000's

# Y2k Agile


Requirements Uncertainty

**Uncertainty** **Certainty**

# Fuzzy vs. Traditional Problems

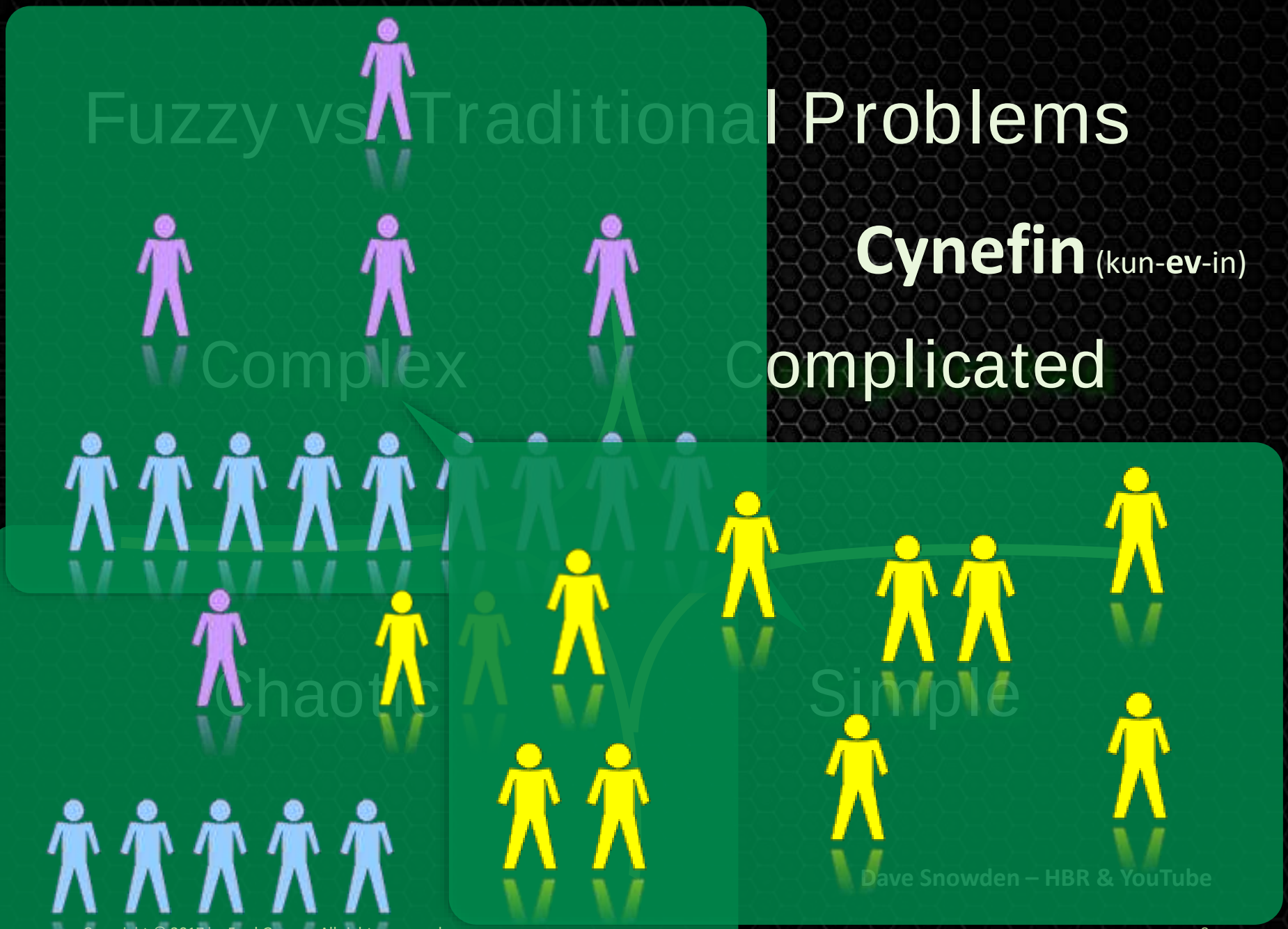



**Cynefin** (kun-**ev**-in)

**Dave Snowden – HBR & YouTube**

# Fuzzy vs. Traditional Problems

**Cynefin** (kun-**ev**-in)



# Requirements Uncertainty

Fuzzy,
but viable


Uncertainty

Certainty

- Assistive technologies abound

# Age of Agents

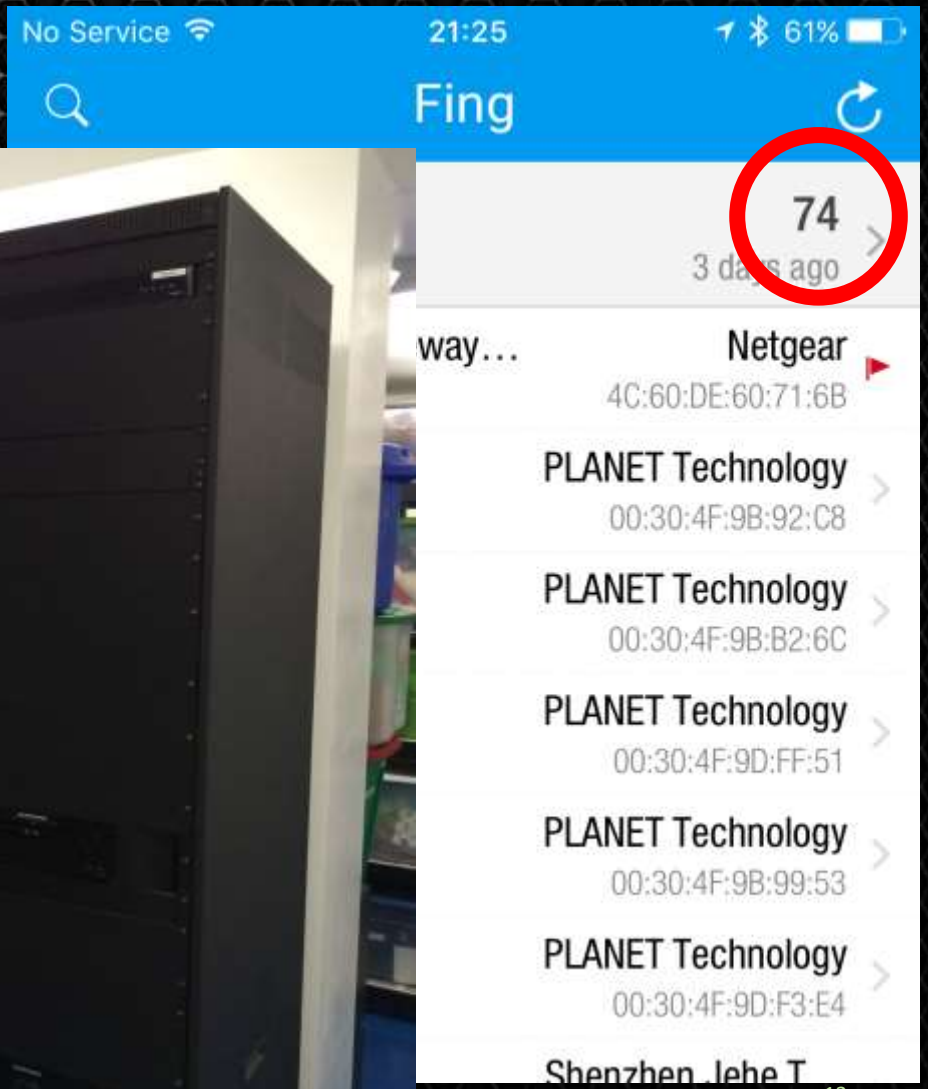
- Google Home, Siri, Alexa, Cortana, …
- Real innovation not voice, but ***interaction*** of ***backing services***

# My Toys...






- What's
- Why B
  - High
  - Bellw
  - ***Fun!***



# Home IoT Hub Contenders

# Amazon Echo

# Google Home

# Xbox One

# Apple TV


# Phillips Hue Bridge


# Samsung Refrigerator

## Are we hitting peak API economy chatter?

Every company loves a good application programming interface. That's a good and bad thing. Enter sprawl, quality issues and API washing in the name of digital transformation.

By Larry Di

Cloud | Data Center | Mobile | Applications | Open Source | Videos | IT Topics | News | Colur

Datamation.com > Mobile/Wireless

Read More in Mobile/Wireless »

# 6 Open Source Middleware Tools for the Internet of Things

Email Article | Comment on this article | Share Articles

Posted December 15, 2015 By Cynthia Harvey

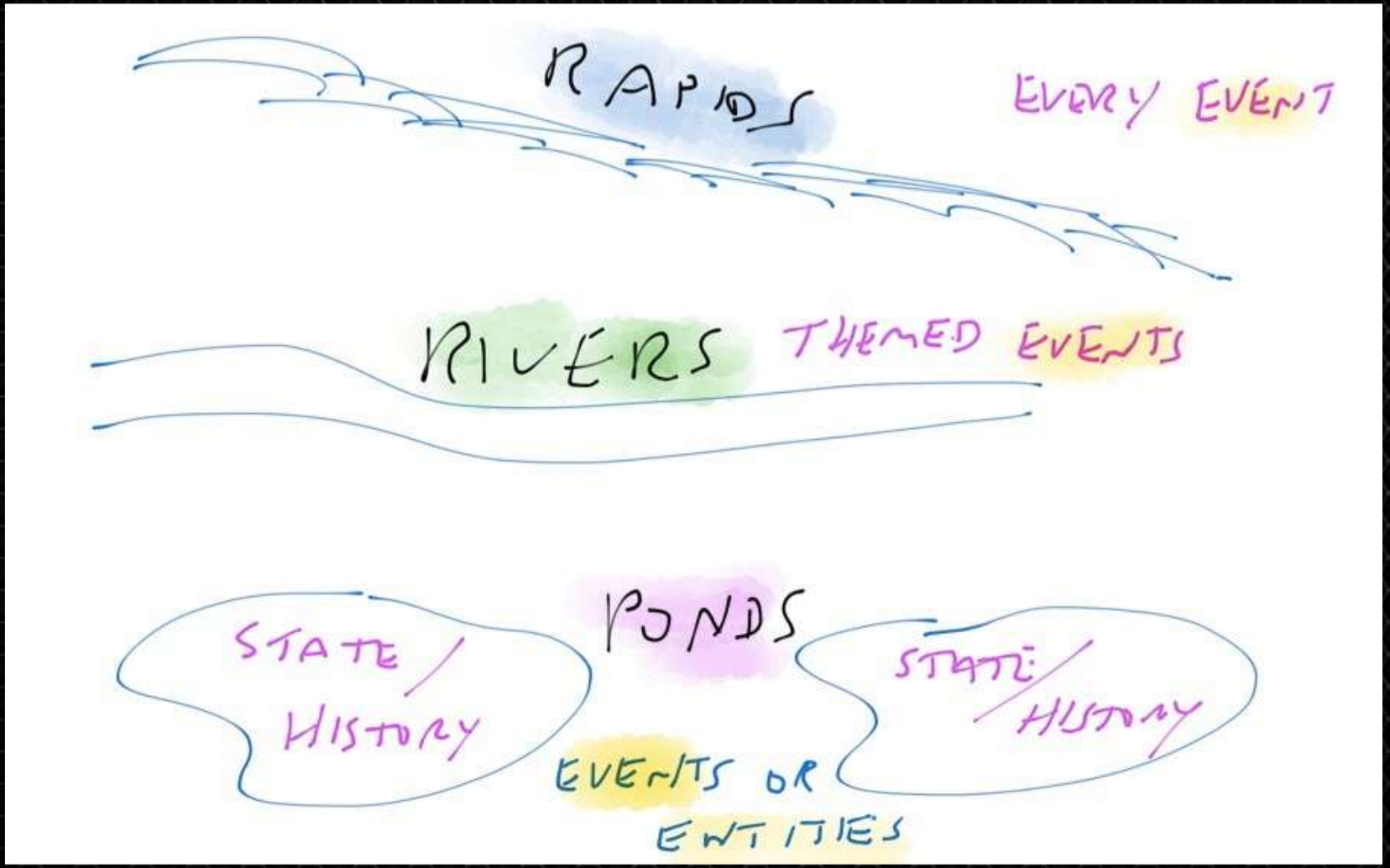
- Uncertainty, so ***Let's Play!***
- Isolate volatility; optimize ***replaceablity***

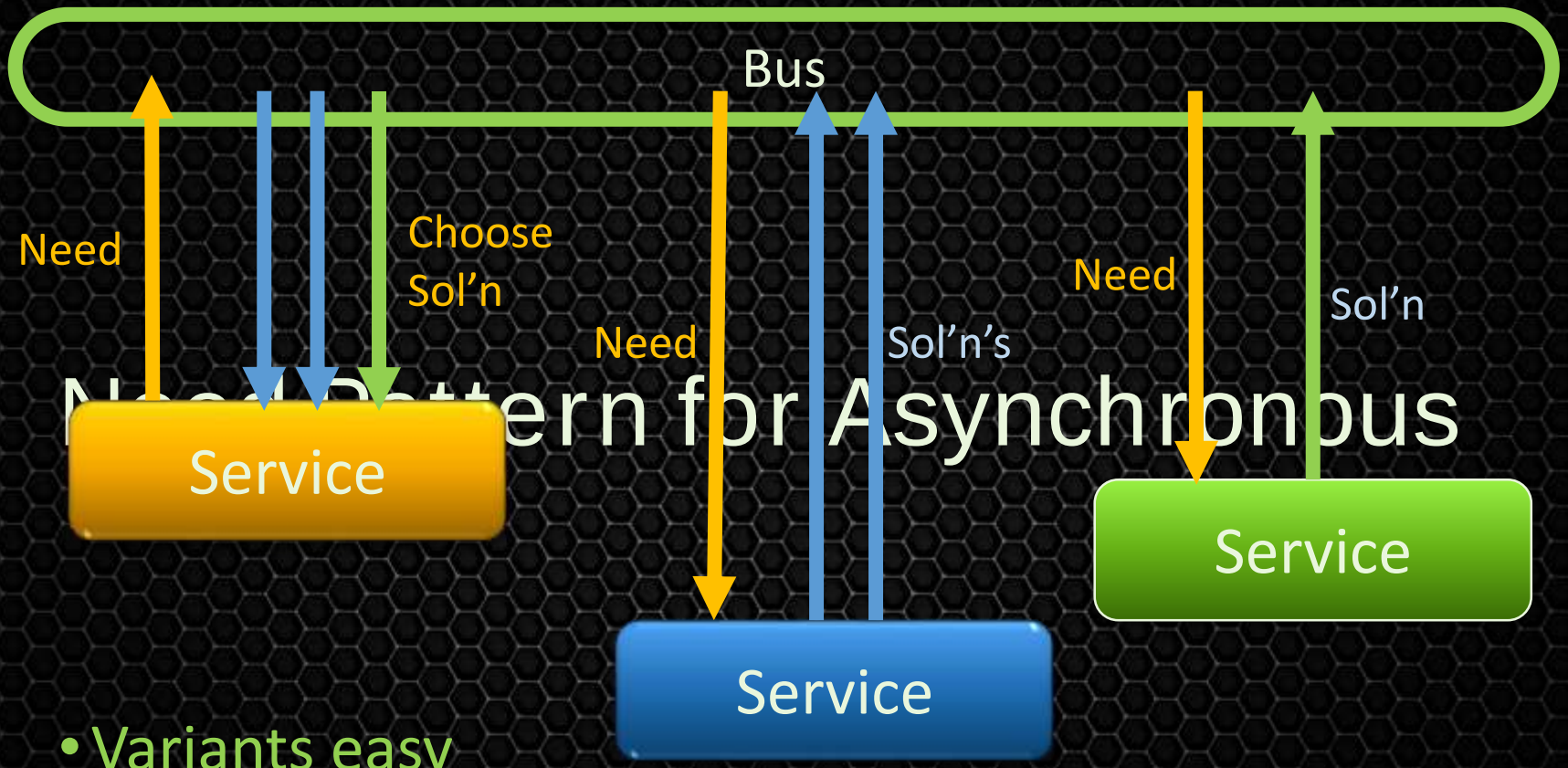
# MicroServices to the Rescue!

  - Changing devices
  - Evolving protocols
- Layer functionality on functionality

# Asynchronous MicroServices

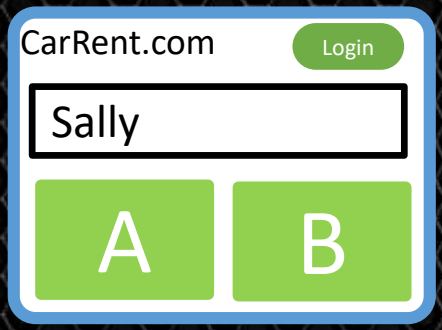
- Less common than ***Synchronous***
  - Consistent with Reactive and Agent-based
- Very, very…
  - Small
  - Loosely coupled
  - Rapidly deployable
- RESTful with registration ***not required***
- Compatible with ***experimentation***






# Need Pattern for Asynchronous

- Variants easy
- Graceful degradation




Car rent

Rental offers

Legacy server

# Incremental Applications

Event Bus

Segmentation

Membership

Location offers

Brand offers

# Technology Choices

- RabbitMQ for event bus (in pub-sub)
  - Simpler than Kafka
  - Good enough
- JSON
  - XML < JSON < compact_binary
- Docker containers; Swarm hosts
- ToDo: Commodity Linux boxes

# Architectural Principles

- Uncertainty => Just-in-Time Design
- Behavior-Oriented Services
  - Services as small as possible with ***kick***
- Publish conclusions
  - If you do something interesting, publish!
- Reliability through:
  - Idempotency – do it again!
  - Redundancy – buses, messages, services, hosts

SampleService.java.java

```
1   public class SampleService implements River.PacketListener {
2
3       public static void main(String[] args) {
4           final RapidsConnection rapidsConnection =
5                   new RabbitMqRapids("sample_service");
6           final River river = new River(rapidsConnection);
7           river.requireValue("source", "phillips_hue_hub");
8           river.require("current_state", "light_count", "version");
9           //river.forbid("key1", "key2");
10          river.register(new Monitor()); // Start receiving traffic
11      }
12  }
```

# Framework from MicroServices Workshop

# Framework from MicroServices Workshop

SampleService.java.java

```java
public class SampleService implements River.PacketListener {

    public static void main(String[] args) {
        final RapidsConnection rapidsConnection =
                new RabbitMqRapids("sample_service");
        final River river = new River(rapidsConnection);
        river.requireValue("source", "phillips_hue_hub");
        river.require("current_state", "light_count", "version");
        //river.forbid("key1", "key2");
        river.register(new Monitor()); // Start receiving traffic
    }
}
```

# Framework from MicroServices Workshop

SampleService.java.java

```java
public class SampleService implements River.PacketListener {

    public static void main(String[] args) {
        final RapidsConnection rapidsConnection =
                new RabbitMqRapids("sample_service");
        final River river = new River(rapidsConnection);
        river.requireValue("source", "phillips_hue_hub");
        river.require("current_state", "light_count", "version");
        //river.forbid("key1", "key2");
        river.register(new Monitor()); // Start receiving traffic
    }
}
```

# Framework from MicroServices Workshop

SampleService.java.java

```java
public class SampleService implements River.PacketListener {

    public static void main(String[] args) {
        final RapidsConnection rapidsConnection =
                new RabbitMqRapids("sample_service");
        final River river = new River(rapidsConnection);
        river.requireValue("source", "phillips_hue_hub");
        river.require("current_state", "light_count", "version");
        //river.forbid("key1", "key2");
        river.register(new Monitor()); // Start receiving traffic
    }
}
```

# Framework from MicroServices Workshop

SampleService.java.java

```
1   public class SampleService implements River.PacketListener {
2
3       public static void main(String[] args) {
4           final RapidsConnection rapidsConnection =
5                   new RabbitMqRapids("sample_service");
6           final River river = new River(rapidsConnection);
7           river.requireValue("source", "phillips_hue_hub");
8           river.require("current_state", "light_count", "version");
9           //river.forbid("key1", "key2");
10          river.register(new Monitor()); // Start receiving traffic
11      }
12  }
```

# Framework from MicroServices Workshop

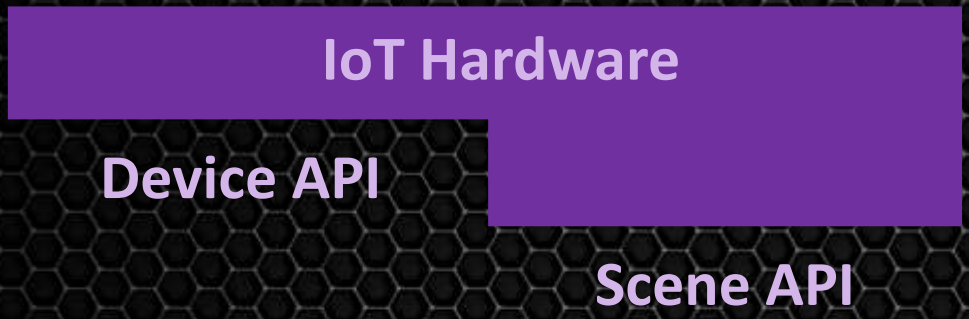
SampleService.java.java

```
1   public class SampleService implements River.PacketListener {
2
3       @Override
4       public void packet(RapidsConnection connection,
5               Packet packet, PacketProblems warnings) {
6           // Process valid messages here, and publish
7           connection.publish(newJsonMessage);
8       }
9
10      @Override
11      public void onError(RapidsConnection connection,
12              PacketProblems errors) {
13          // Optionally process invalid messages here
14      }
15  }
```

# Framework from MicroServices Workshop

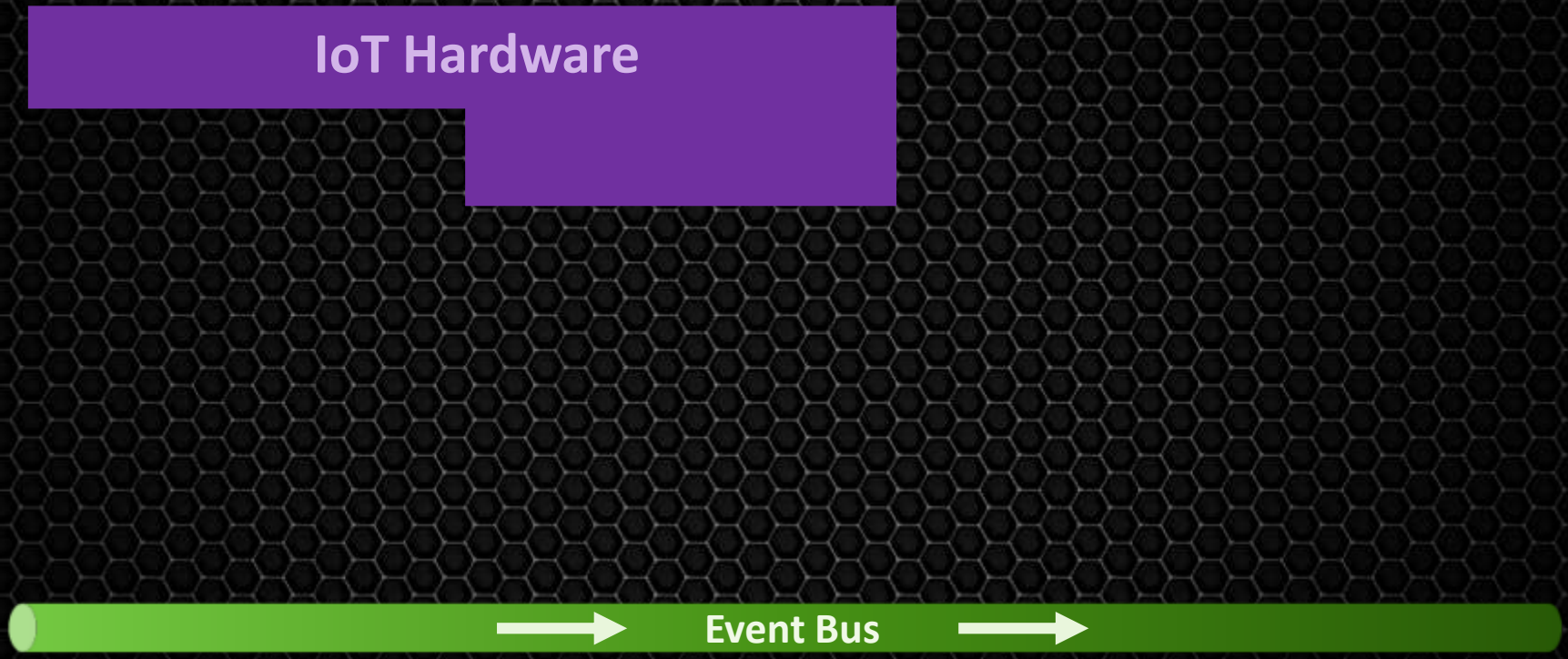
SampleService.java.java

```
1   public class SampleService implements River.PacketListener {
2
3       @Override
4       public void packet(RapidsConnection connection,
5               Packet packet, PacketProblems warnings) {
6           // Process valid messages here, and publish
7           connection.publish(newJsonMessage);
8       }
9
10      @Override
11      public void onError(RapidsConnection connection,
12              PacketProblems errors) {
13          // Optionally process invalid messages here
14      }
15  }
```

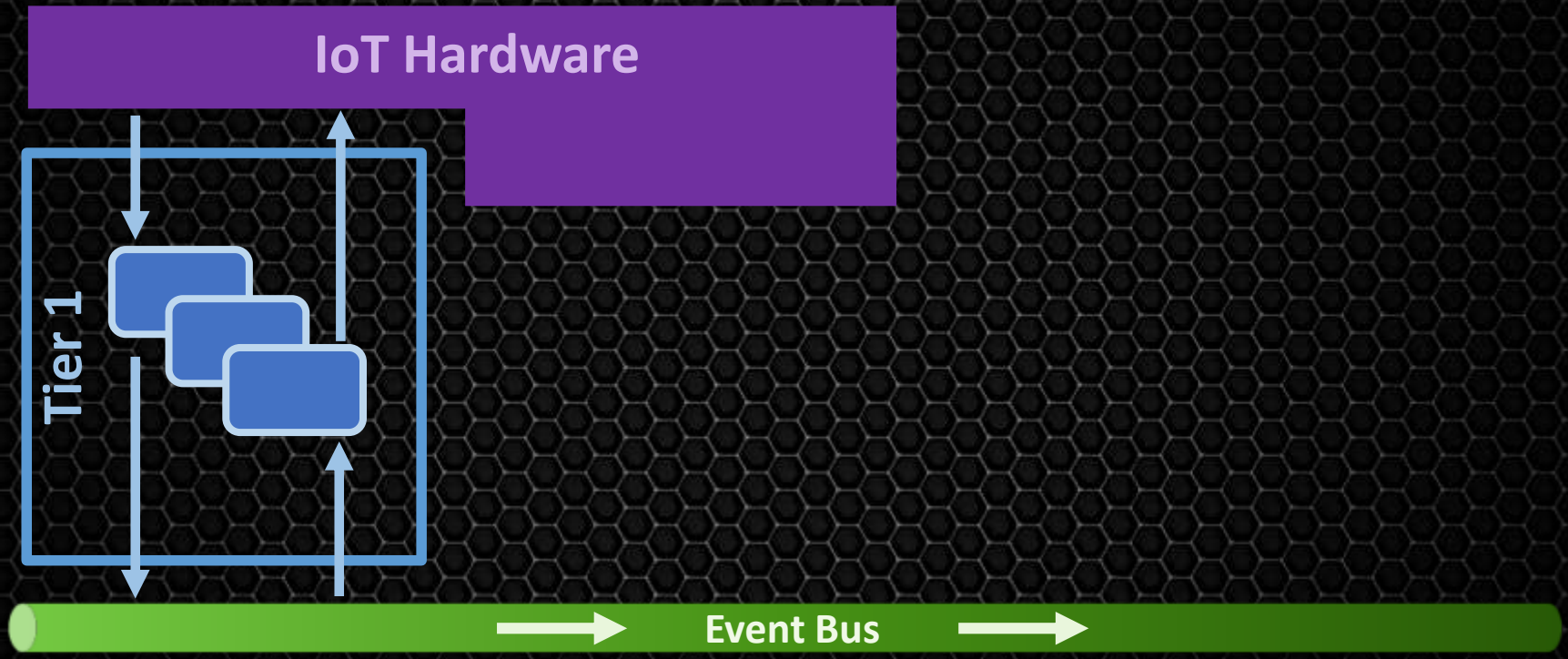



# Architecture: Hardware

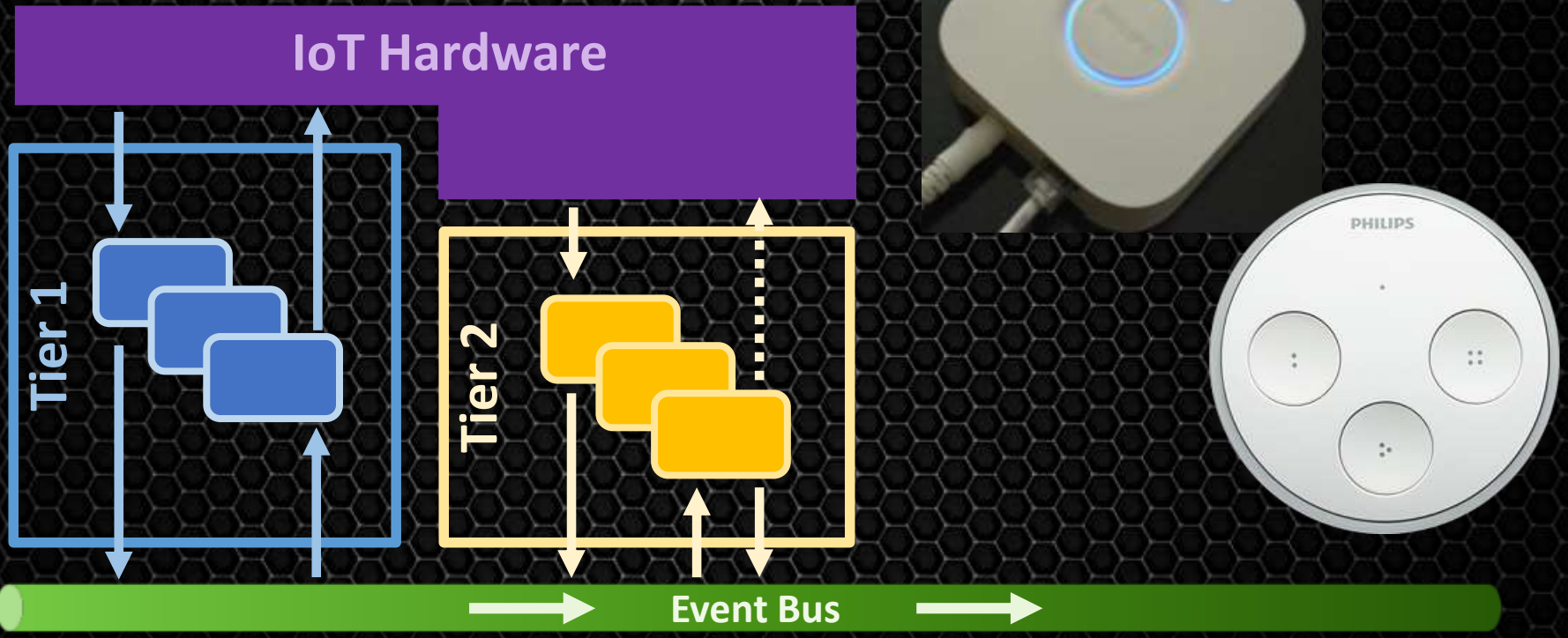
# Add Event Bus



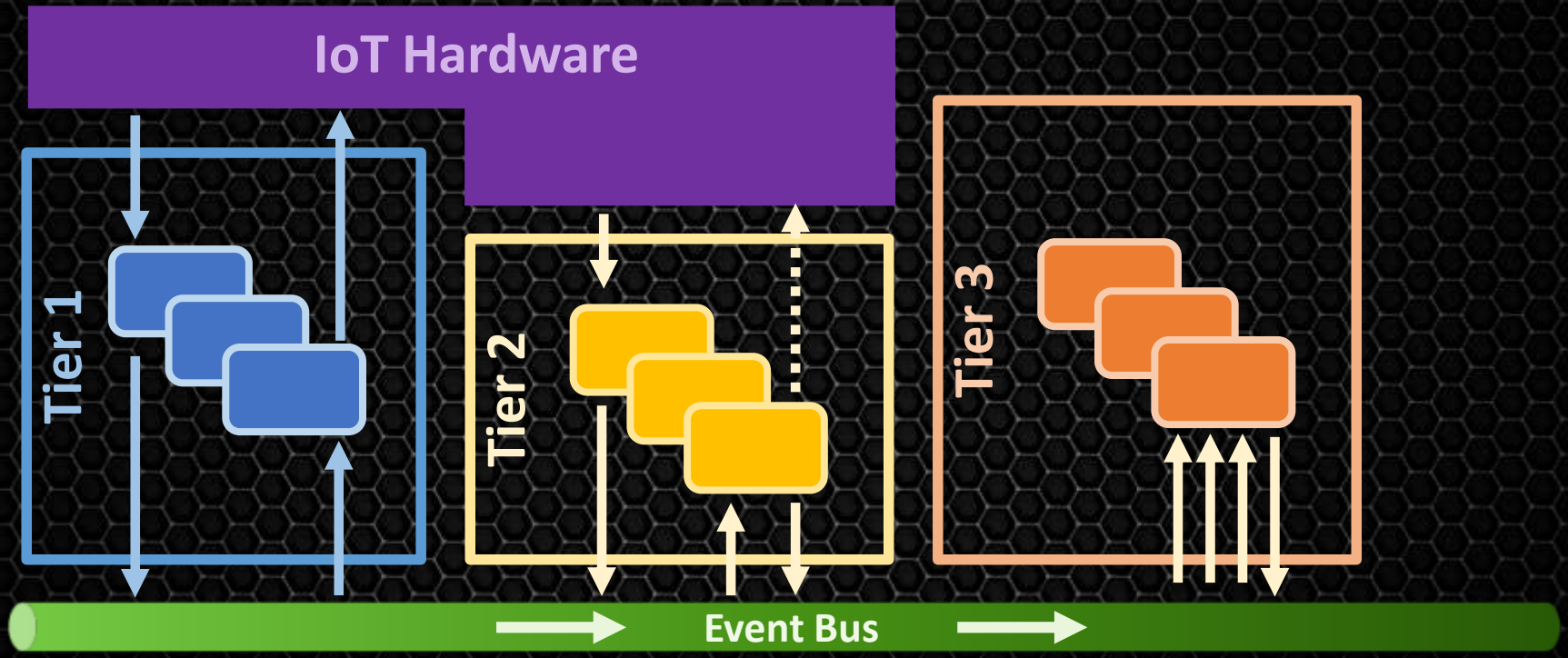
# Tier 1: Hardware Access



# Tier 2: Scenes

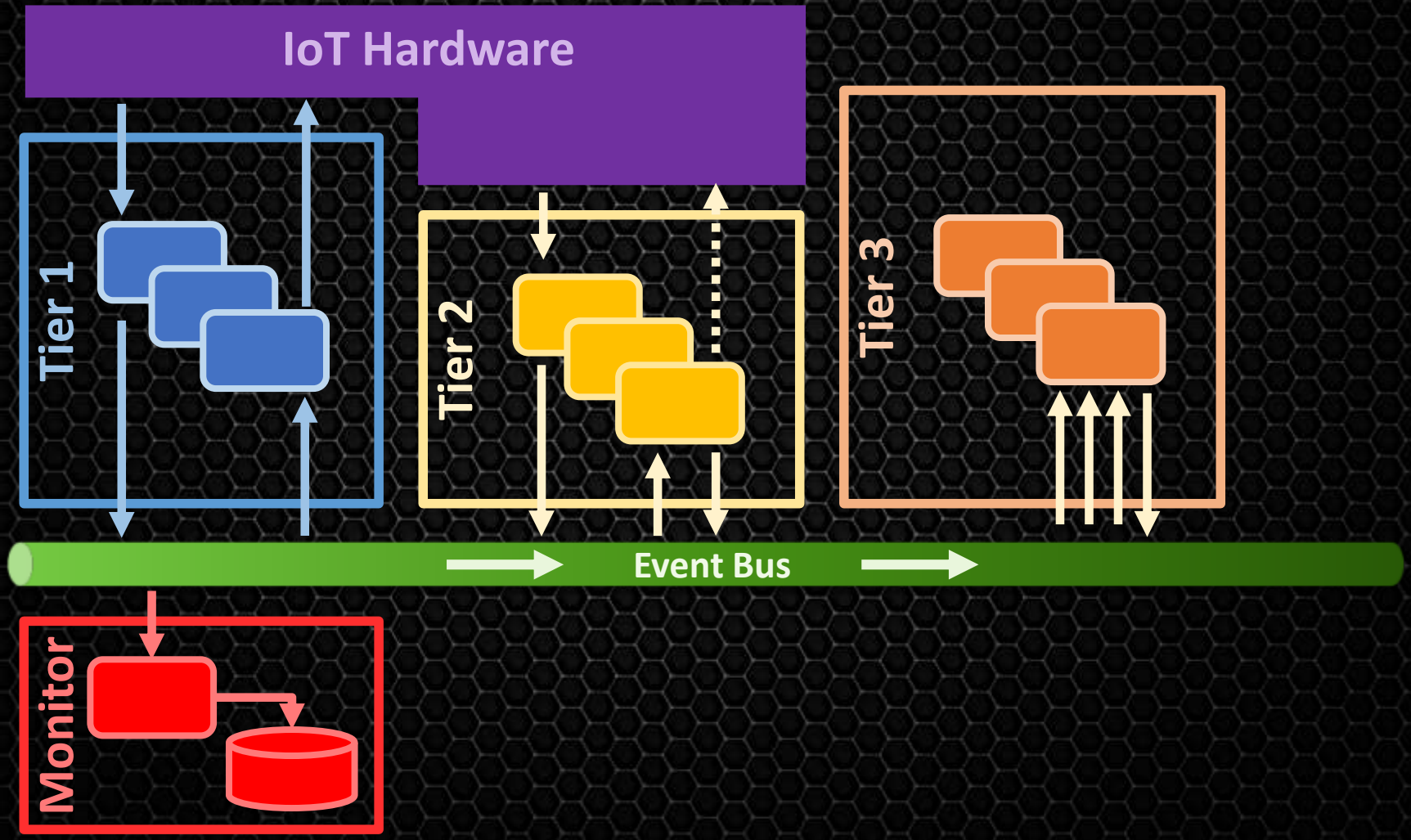


# Tier 3: Reactive Services

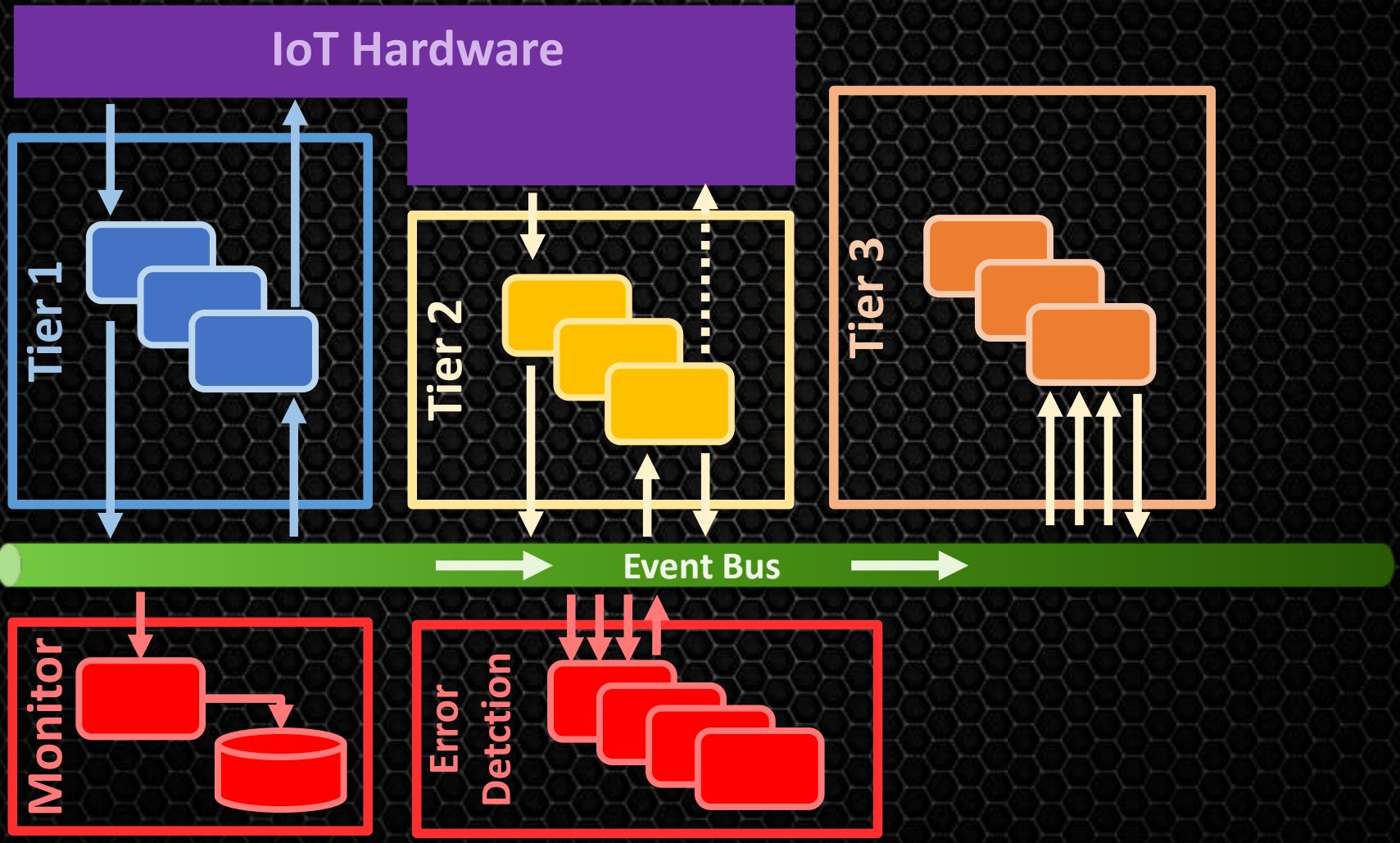



e.g., **ToD + Motion => Set Scene**

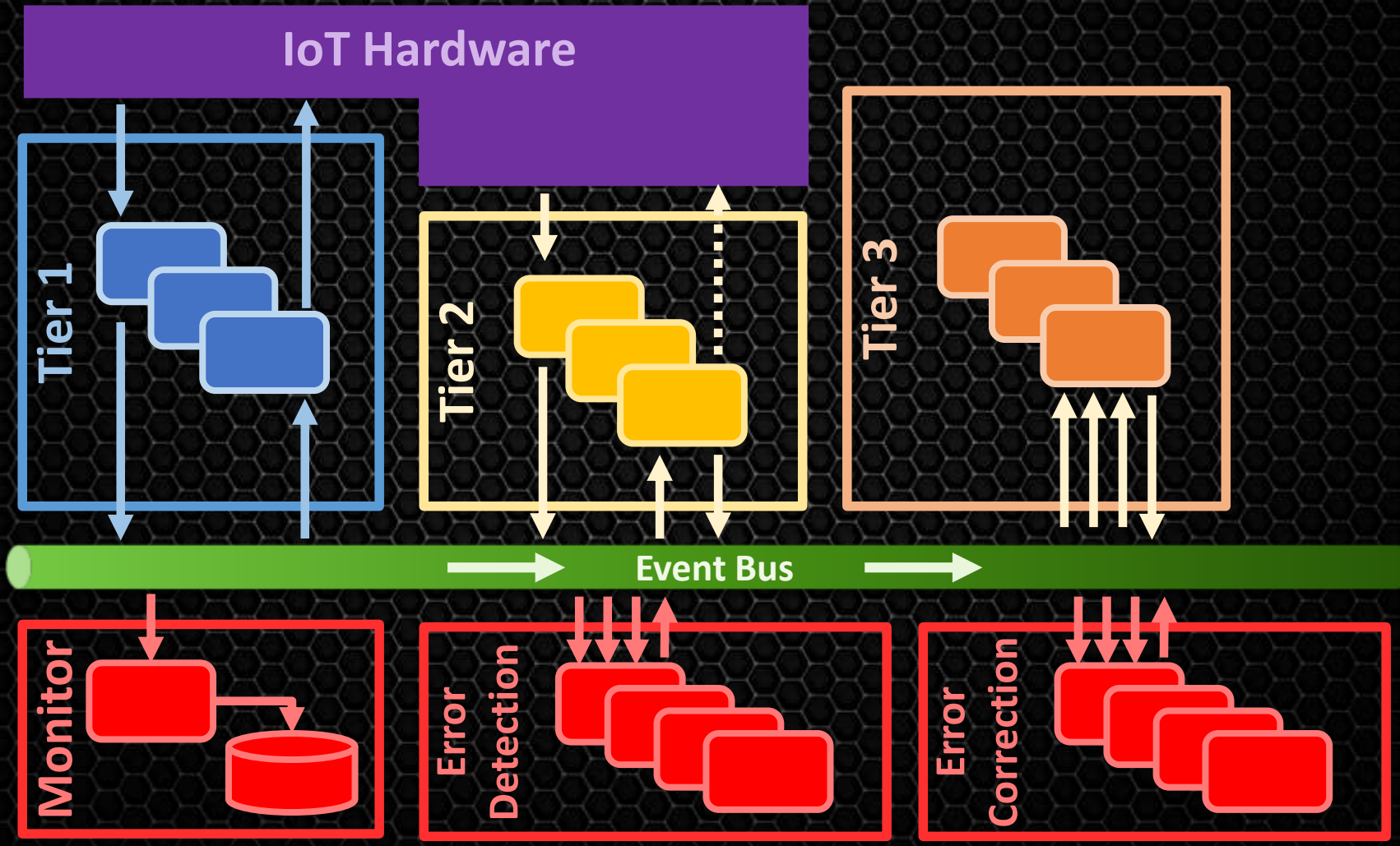
# MetaServices: Monitor



# MetaServices: Detection

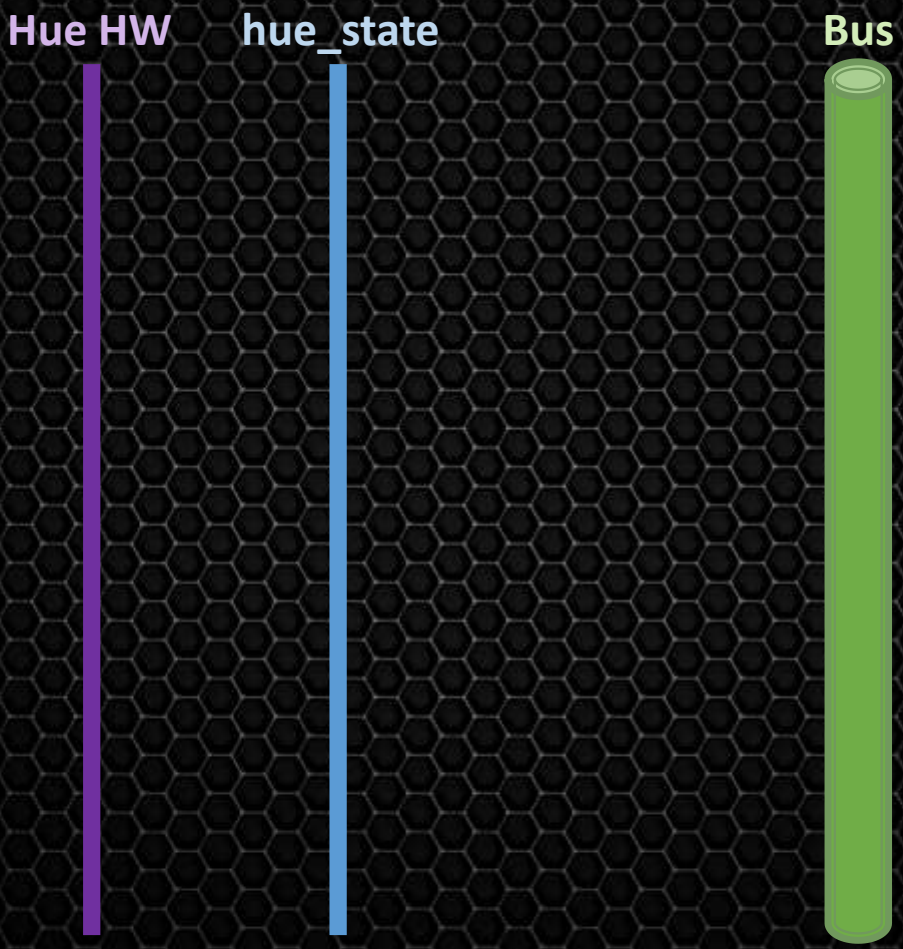


# MetaServices: Correction

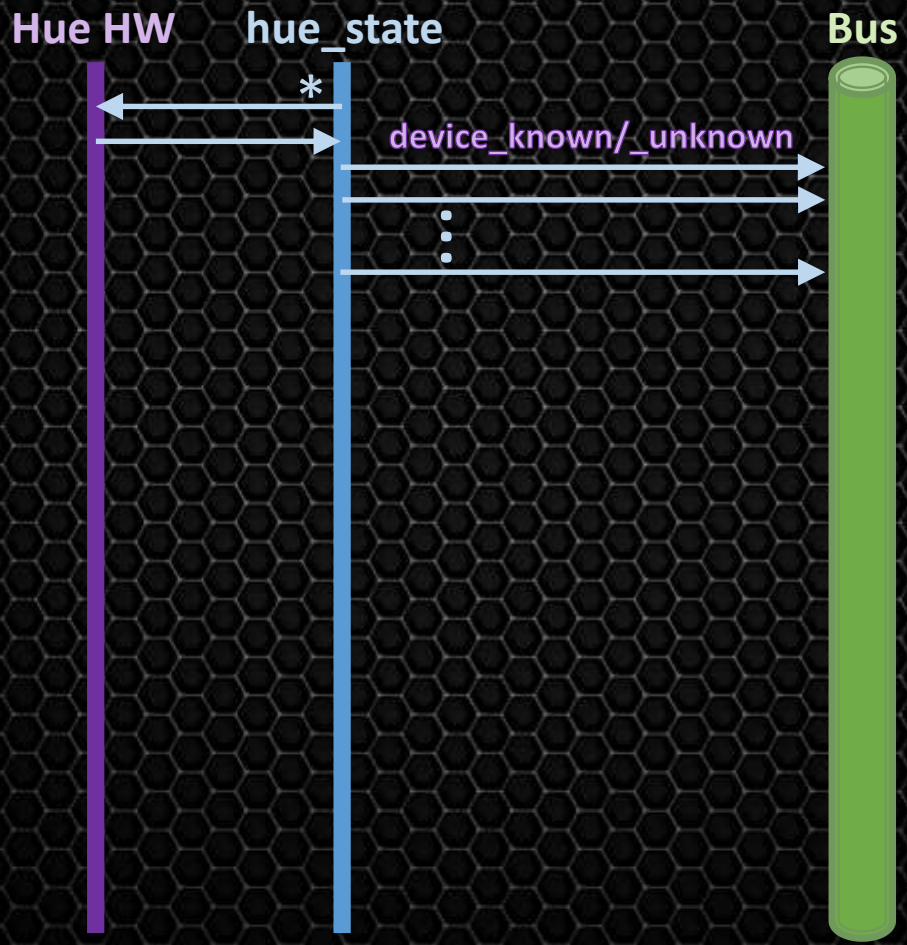



# Choreography – Sticky Colors

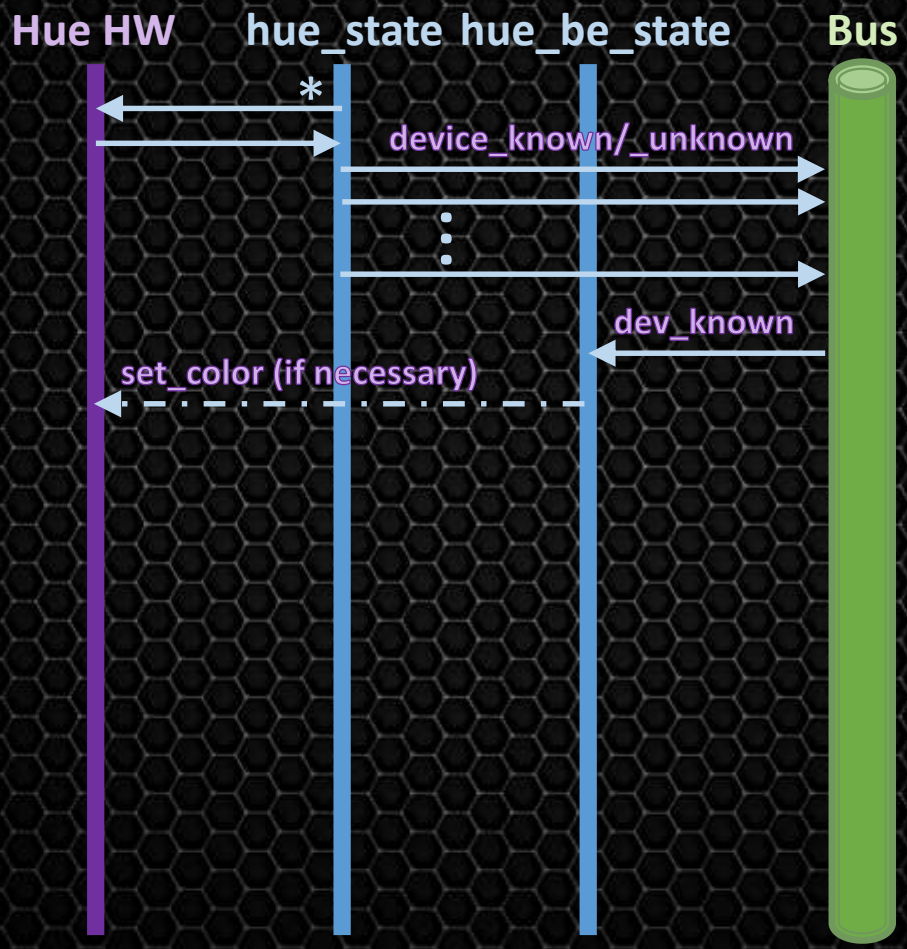
# Choreography – Sticky Colors



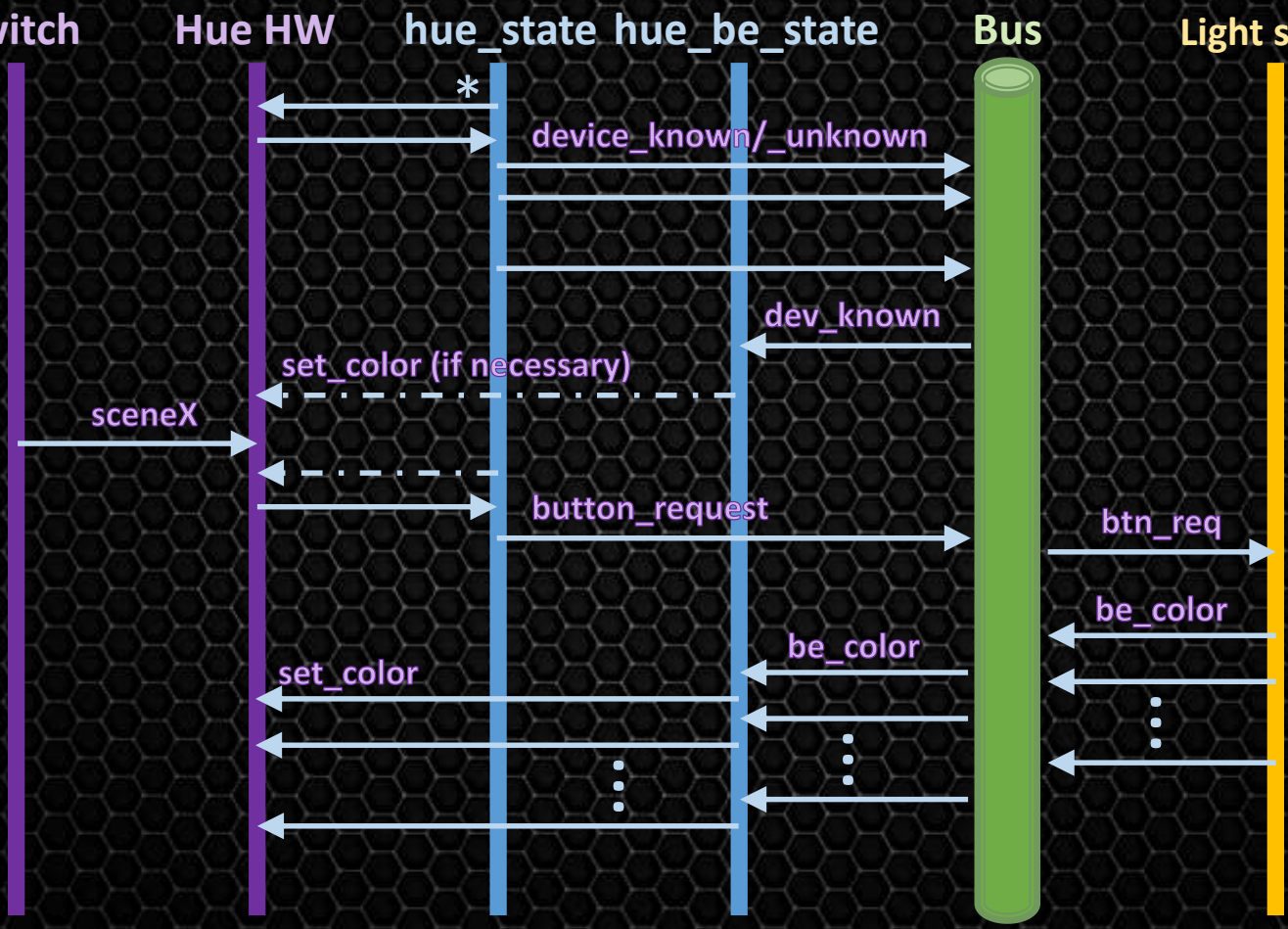
# Choreography – Sticky Colors



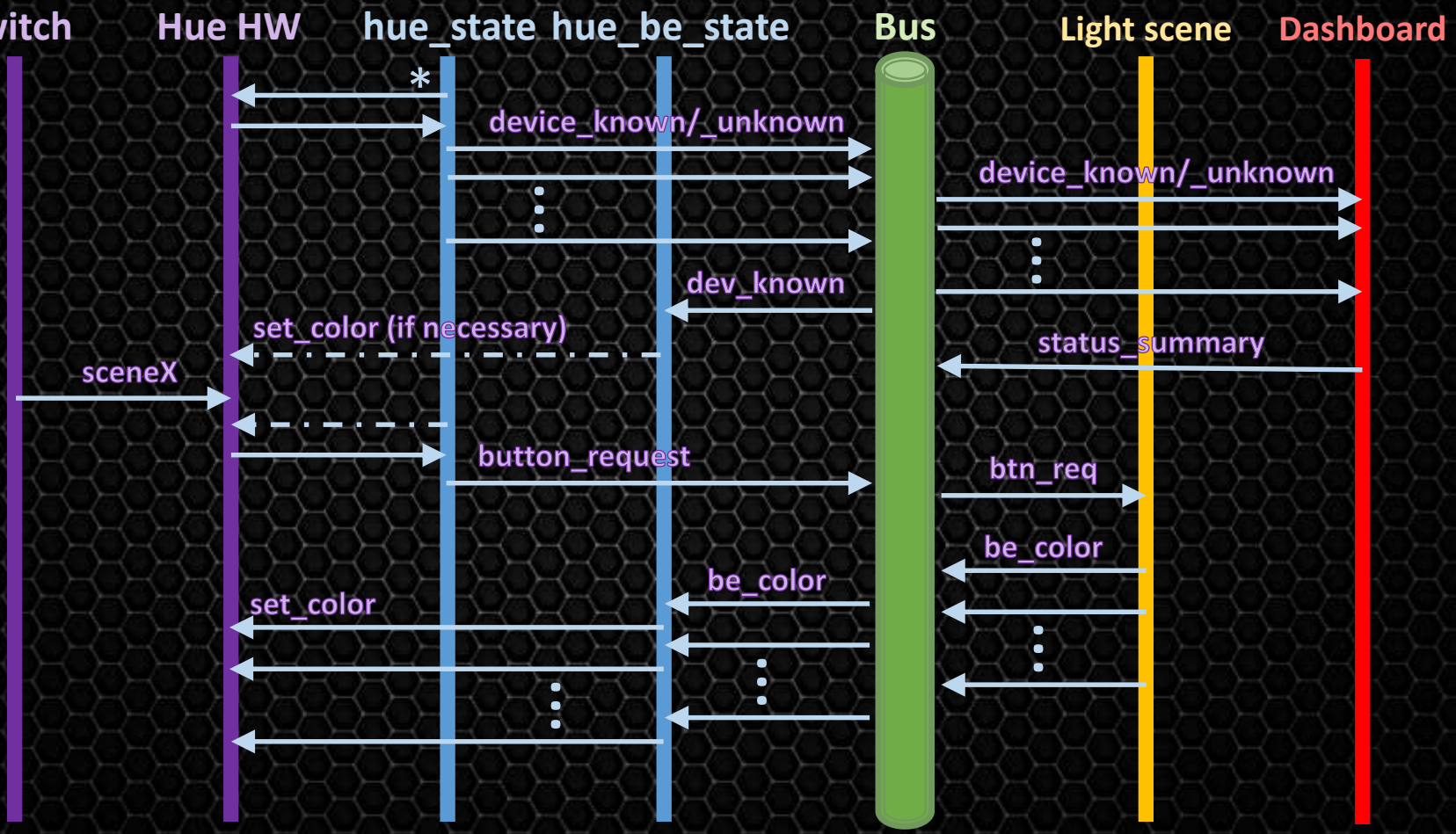
# Choreography – Sticky Colors



# Choreography – Sticky Colors



# Choreography – Sticky Colors



# Observations

- Lots of messages
- No RESTful interfaces between services
- No constraints on multiple instances/versions
- Unit tests not required
- Fast-failing system rather than acceptance tests

# XP Values at Work

- Feedback
- Communication
- Simplicity
- Courage
- Respect

# Agile Practices and Roles

- Stand ups
- Story narratives
- Retrospectives
- Estimates
- Iterations
- Mandatory pairing

- Unit tests
- Acceptance tests
- Refactoring
- Patterns
- Continuous integration

- BA
- Developer
- Architect
- SCRUM master
- QA
- Manager
- Customer

# Fuzzy

## ~~Agile~~ Practices and Roles

- ~~Stand ups~~
- ~~Story narratives~~
- ~~Retrospectives~~
- ~~Estimates~~
- ~~Iterations~~
- ~~Mandatory~~ pairing
- ~~Unit tests~~
- ~~Acceptance tests~~
- ~~Refactoring~~
- ~~Patterns~~
- ~~Continuous integration~~
- ~~BA~~
- Developer
- ~~Architect~~
- ~~SCRUM master~~
- ~~QA~~
- ~~Manager~~
- Customer

# Requirements Uncertainty

**Fuzzy,**
**but viable**

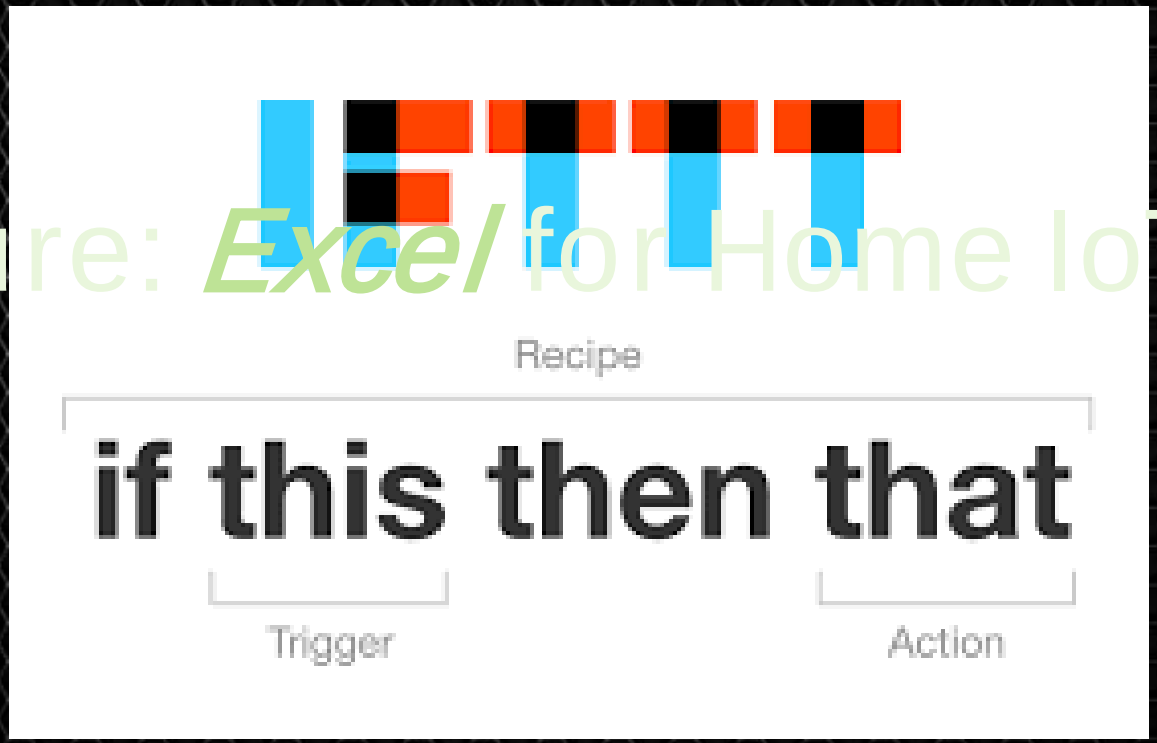

**Uncertainty** — **Certainty**

- Idea-focused features
- Full-stack developers
- Fast failure systems
- MicroServices
- Event-based architecture
- Continuous deployment

- Stories
- Specialists
- TDD
- Acceptance tests
- Migration scripts

# More Devi

# Future: *Excel* for Home IoT:



ker 1.12

# MicroServices, IoT, and **Agile… Seriously!**

Fred George

fredgeorge@acm.org

@fgeorge52