Programming Languages to Runtime Execution: A Continuum

Red Hat Research Day Europe
15/09/2022

Frédéric Le Mouël*, Patrik Fortier*, Arthur Navarro*°, Julien Ponge°,
Clément Escoffier°

* University of Lyon, INSA Lyon, Inria
° Red Hat

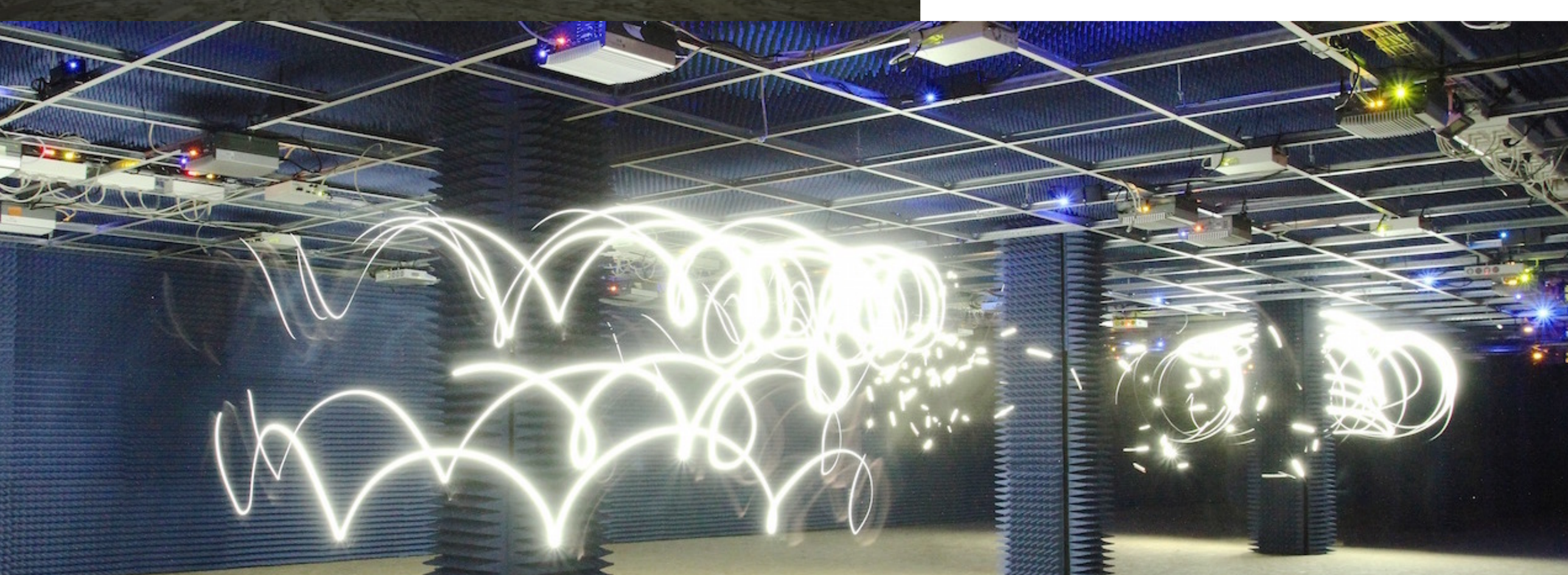
Lyon, France ?

- 2h from Paris by train
- 1.4M inhabitants
- 2nd economical region in France
- Region, famous for:
  - Gastronomy
  - Metallurgy sector
  - Chemical sector
  - Healthcare, pharmaceutic sector
  - Electric, electronic sector
  - IT services sector

CITI Lab

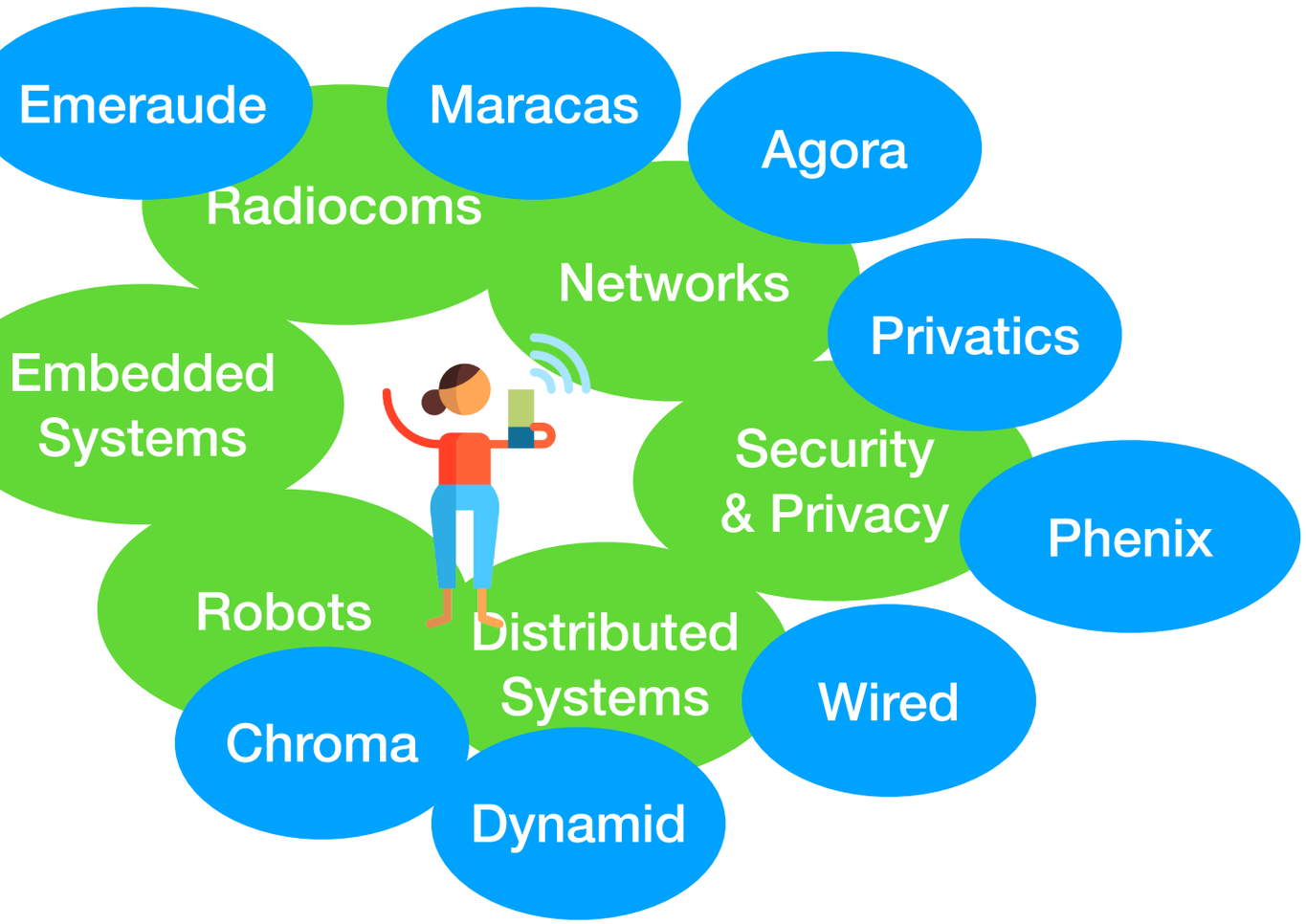
- Topic: Telecommunications
- ~130 members (~40 professors), 8 teams
- Common Labs: Nokia Bell, Orange - Chair: SPIE ICS
- 12 plateforms : EquipEx FIT/CorteXLab, EquipEx+ TIRREX
- 3 spinoff startups
- 9.6M€ turnover, 4.3M€ revenue generated in 2022

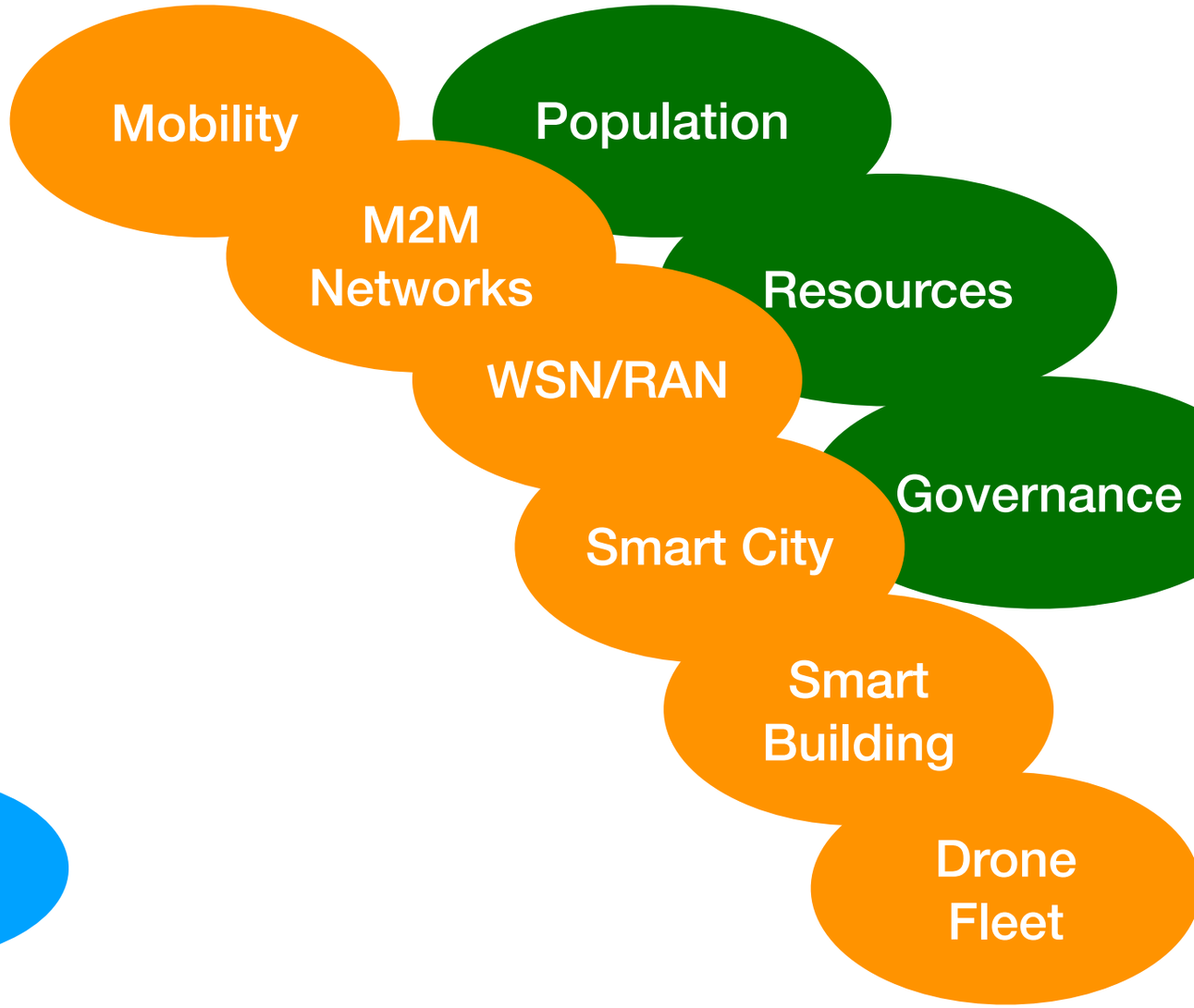
CorteXLab Platform



Scientific project
« Reconnected Society »

citi lab





Scientific project « Reconnected Society »

citi lab

Population

Mobility

M2M Networks

Resources

WSN/RAN

Governance

Smart City

Smart Building

Drone Fleet

Complex Systems Modelling

Large-scale Systems

Collaborative Systems

Emeraude

Maracas

Agora

Radiocoms

Networks

Privatics

Embedded Systems

Security & Privacy

Phenix

Robots

Distributed Systems

Wired

Chroma

Dynamid

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Inria

CPE LYON

GRAME CENTRE NATIONAL DE CRÉATION MUSICALE

UNIVERSITÉ DE LYON

Scientific project
« Reconnected Society »

citi lab

Population

Mobility

M2M Networks

Resources

WSN/RAN

Governance

Smart City

Smart Building

Drone Fleet

Complex Systems Modelling

Deep/Reinf Learning

Quantum Comm

Large-scale Systems

Collaborative Systems

Emeraude

Maracas

Agora

Radiocoms

Networks

Privatics

Embedded Systems

Security & Privacy

Phenix

Robots

Distributed Systems

Wired

Chroma

Dynamid

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Inria

CPE LYON

GRAME CENTRE NATIONAL DE CRÉATION MUSICALE

UNIVERSITÉ DE LYON

Scientific project
« Reconnected Society »

citi lab

Population

Mobility

M2M Networks

Resources

WSN/RAN

Governance

Smart City

Smart Building

Drone Fleet

Complex Systems Modelling

Deep/Reinf Learning

Quantum Comm

Large-scale Systems

IoT

6G

Fog/Edge

Zero-Energy IoT

Collaborative Systems

Emeraude

Maracas

Agora

Radiocoms

Networks

Privatics

Embedded Systems

Security & Privacy

Phenix

Robots

Distributed Systems

Wired

Chroma

Dynamid

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Inria

CPE LYON

GRAME CENTRE NATIONAL DE CRÉATION MUSICALE

UNIVERSITÉ DE LYON

Scientific project « Reconnected Society »

citi lab

Population

Mobility

M2M Networks

Resources

WSN/RAN

Governance

Smart City

Smart Building

Drone Fleet

Complex Systems Modelling

Deep/Reinf Learning

Quantum Comm

Large-scale Systems

IoT

6G

Fog/Edge

Zero-Energy IoT

Collaborative Systems

Blockchain

Carbon-Neutral Digital Systems

Autonomous Robots

Regulatory-compliant Systems

Emeraude

Maracas

Agora

Radiocoms

Networks

Privatics

Embedded Systems

Security & Privacy

Phenix

Robots

Distributed Systems

Wired

Chroma

Dynamid

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Inria

CPE LYON

GRAME CENTRE NATIONAL DE CRÉATION MUSICALE

UNIVERSITÉ DE LYON

Frédéric Le Mouël

- Professor, head of CITI Lab
- Chair in Internet of Things
- Chair in Edge AI
- Topics: Distributed Systems, Edge/Fog Computing
- Industrial use cases: IT sector (SPIE ICS, Red Hat), Automotive sector (VALEO, Stellantis)
- Twitter: @flemouel

Home Automation

Mobile Cloud Computing

Green Buildings

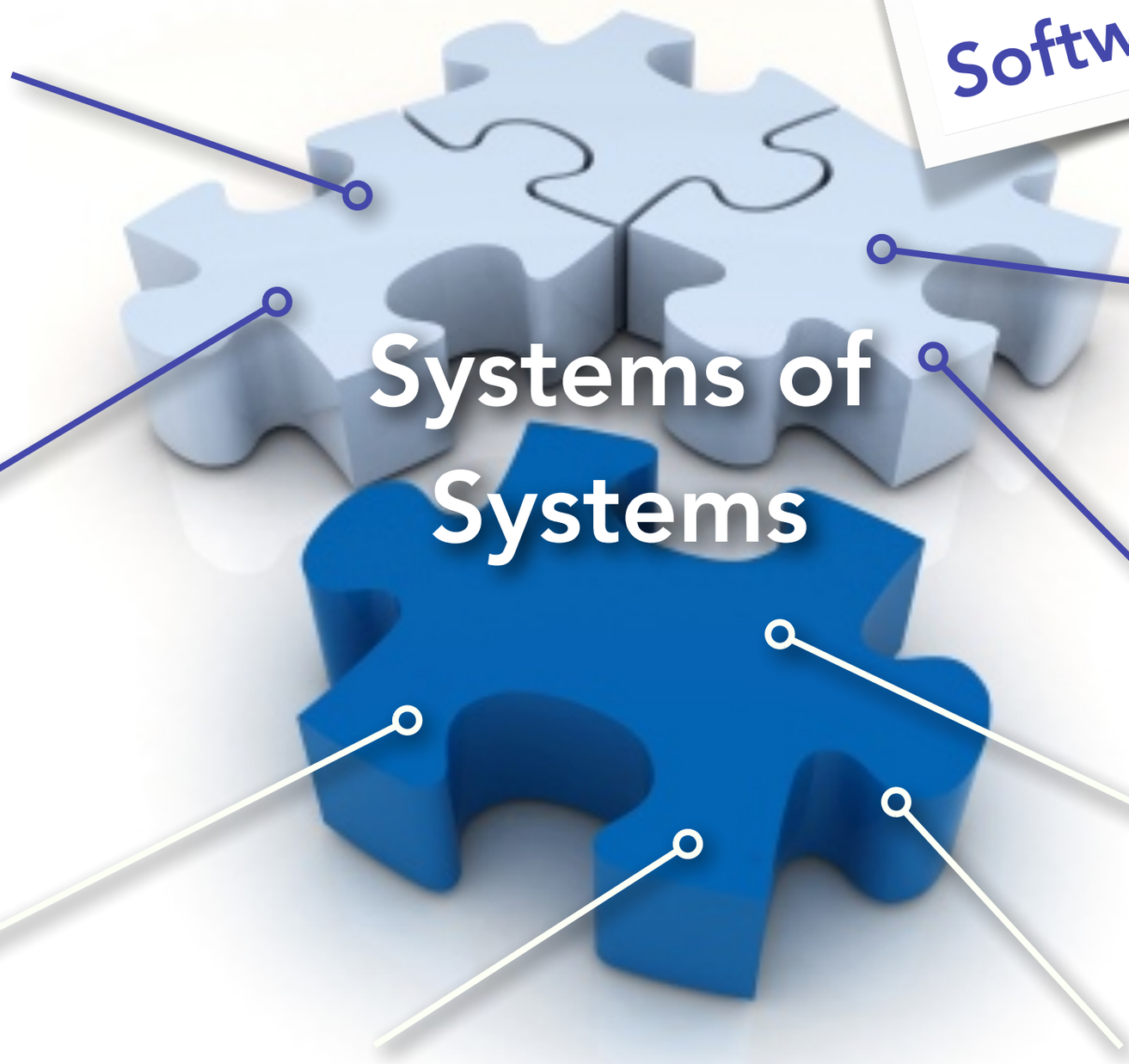
Autonomous Robotics

Smart Cities

Smart Cars

Internet/Web of Things

Integrated Software Systems



Multi-agents Systems

Distributed Systems

Operating Systems

Embedded Mobile Code

Middleware & Platforms

Algorithms

Optimization A.I.

Programming Languages, DSL

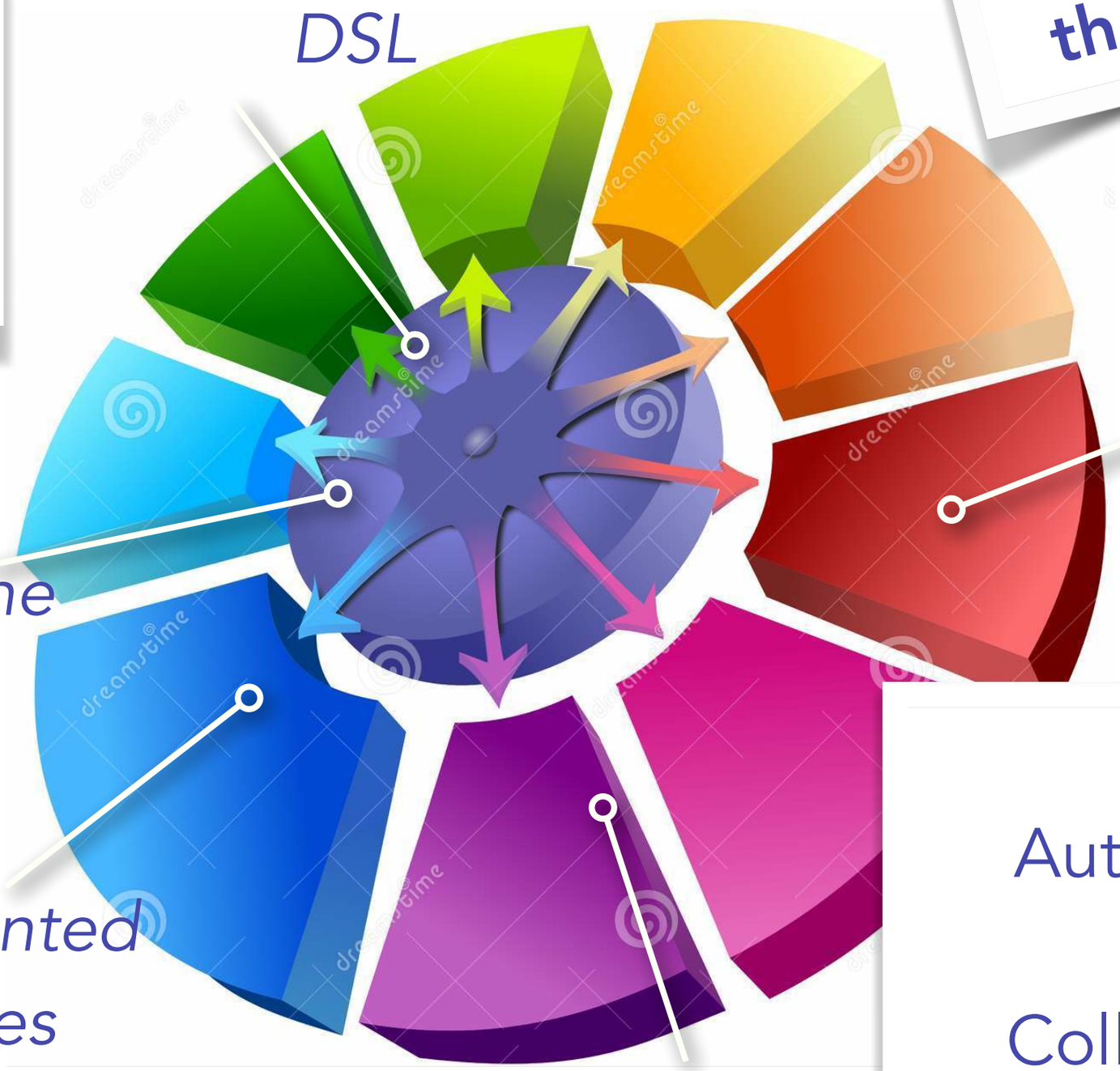
I
Adaptive
in the Small

Programming the Move !

Context-aware DSL

Distributed Decision

Adaptive Virtual Machine

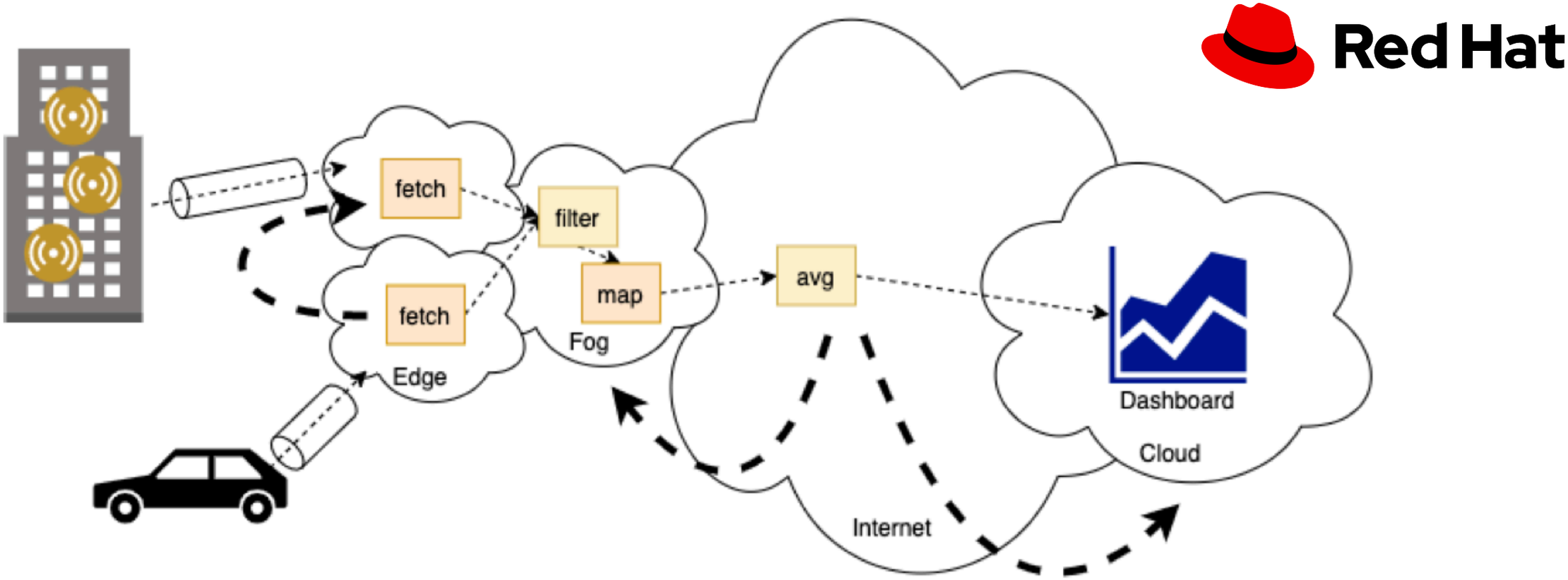


Long-life Software-Oriented Architectures

Bio-inspired Algorithms

II
Autonomous
and
Collaborative
in the Large

Automatic Service Deployment



Moving with your services and …
what if you switch from a Kubernetes infrastructure to a Mesos one?

Automatic Service Deployment

- Proposal: a DSL to manage and automatise your deployment
- Service generation according to the targeted infrastructure
- Synchronization avoiding SPOF

Abstraction	Description
@Service	Services are defined like function annotated. Dyninka generates a service with the same capabilities than the function. The annotation @Service is used to identify functions from which we generate services.
@Placed(val location: String)	User-defined local variable. The annotation *@Placed(val location:String)* is used to place variables. This technique is used for introducing local memory to normally stateless functions and services.
Abstracted synchronization	Generated services provides asynchronous constructs (Promise, Future) to perform dead-lock free communication.
Client side type safety	*T :->Future<T>* Type safety performed on client side is respected when converting functions into asynchronous services.

Automatic Service Deployment

Kotlin (prog lang)
Gradle (deps)
ANTLR (parsing)

```
@Placed("get") val vertx: Vertx = Vertx.vertx()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject>
{
  return WebClient.create(vertx)
    .get("sensor","/").expect(SC_OK)
    .send().map {
      it.bodyAsJsonObject()
    }
}
```

[...]

```
@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject>
{
  val promise = Promise.promise<JsonObject>()
  this.onSuccess {
    val next = it.getFloat("Value")
    movingAvr = (next + counter * movingAvr) / (counter + 1)
    counter++
    val result = JsonObject().put("Average", movingAvr)
    promise.complete(result)
  }
  return promise.future()
}

fun main()
{
  get().filter().map().movingAvg()
}
```

Automatic Service Deployment

```
@Placed("get") val vertx: Vertx = Vertx.vertx()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject>
{
  return WebClient.create(vertx)
    .get("sensor","/").expect(SC_OK)
    .send().map {
      it.bodyAsJsonObject()
    }
}
```

[...]

```
@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject>
{
  val promise = Promise.promise<JsonObject>()
  this.onSuccess {
    val next = it.getFloat("Value")
    movingAvr = (next + counter * movingAvr) / (counter + 1)
    counter++
    val result = JsonObject().put("Average", movingAvr)
    promise.complete(result)
  }
  return promise.future()
}

fun main()
{
  get().filter().map().movingAvg()
}
```

Automatic Service Deployment

```
@Placed("get") val vertx: Vertx = Vertx.vertx()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject>
{
  return WebClient.create(vertx)
    .get("sensor","/").expect(SC_OK)
    .send().map {
      it.bodyAsJsonObject()
    }
}
```

[...]

```
@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject>
{
  val promise = Promise.promise<JsonObject>()
  this.onSuccess {
    val next = it.getFloat("Value")
    movingAvr = (next + counter * movingAvr) / (counter + 1)
    counter++
    val result = JsonObject().put("Average", movingAvr)
    promise.complete(result)
  }
  return promise.future()
}

fun main()
{
  get().filter().map().movingAvg()
}
```

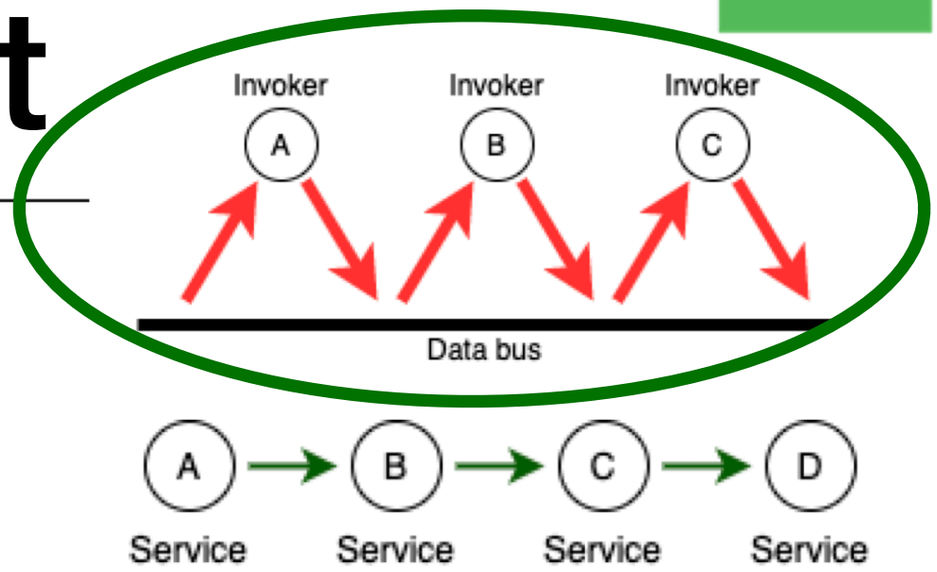
Automatic Service Deployment

```
@Placed("get") val vertx: Vertx = Vertx.vertx()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject>
{
  return WebClient.create(vertx)
    .get("sensor","/").expect(SC_OK)
    .send().map {
      it.bodyAsJsonObject()
    }
}
```

[...]

```
@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject>
{
  val promise = Promise.promise<JsonObject>()
  this.onSuccess {
    val next = it.getFloat("Value")
    movingAvr = (next + counter * movingAvr) / (counter + 1)
    counter++
    val result = JsonObject().put("Average", movingAvr)
    promise.complete(result)
  }
  return promise.future()
}

fun main()
{
  get().filter().map().movingAvg()
}
```

Automatic Service Deployment

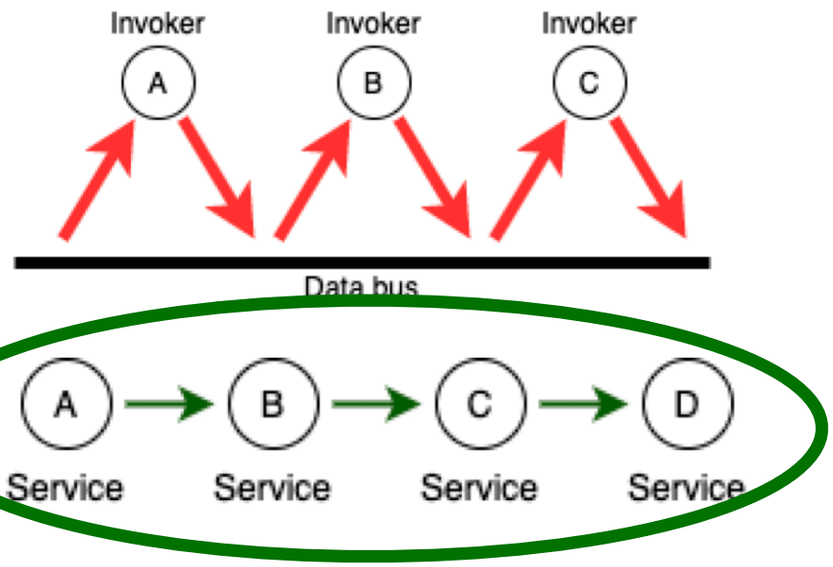



```
@Placed("get") val vertx: Vertx = Vertx.vertx()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject>
{
  return WebClient.create(vertx)
    .get("sensor","/").expect(SC_OK)
    .send().map {
      it.bodyAsJsonObject()
    }
}
```

[...]

```
@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject>
{
  val promise = Promise.promise<JsonObject>()
  this.onSuccess {
    val next = it.getFloat("Value")
    movingAvr = (next + counter * movingAvr) / (counter + 1)
    counter++
    val result = JsonObject().put("Average", movingAvr)
    promise.complete(result)
  }
  return promise.future()
}

fun main()
{
  get().filter().map().movingAvg()
}
```

Automatic Service Deployment

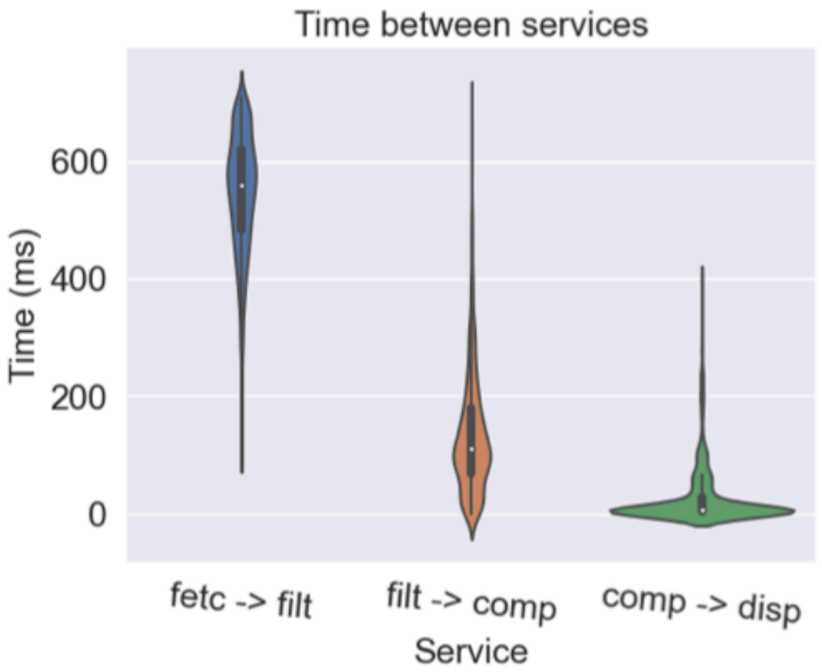



```
@Placed("get") val vertx: Vertx = Vertx.vertx()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject>
{
  return WebClient.create(vertx)
    .get("sensor","/").expect(SC_OK)
    .send().map {
      it.bodyAsJsonObject()
    }
}
```

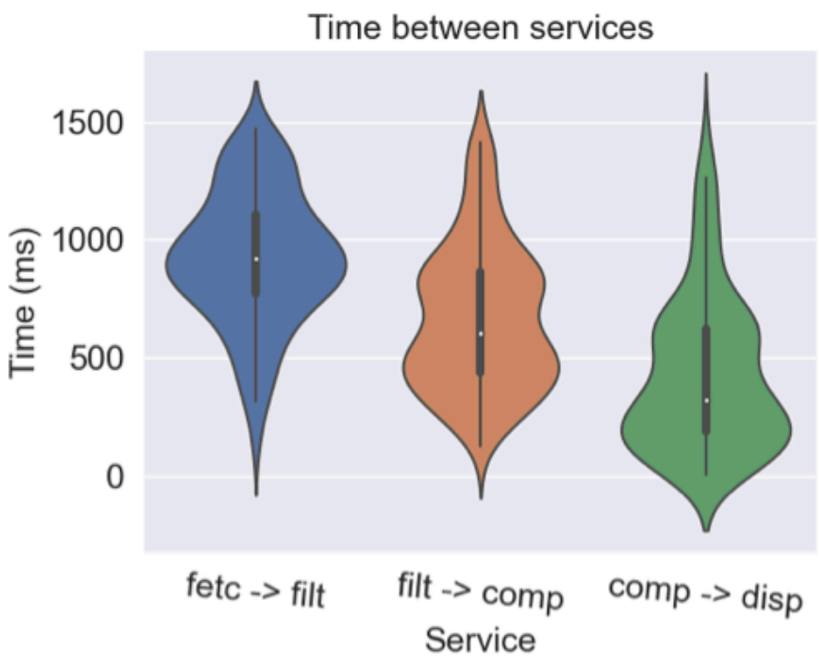
[...]

```
@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject>
{
  val promise = Promise.promise<JsonObject>()
  this.onSuccess {
    val next = it.getFloat("Value")
    movingAvr = (next + counter * movingAvr) / (counter + 1)
    counter++
    val result = JsonObject().put("Average", movingAvr)
    promise.complete(result)
  }
  return promise.future()
}

fun main()
{
  get().filter().map().movingAvg()
}
```

Automatic Service Deployment



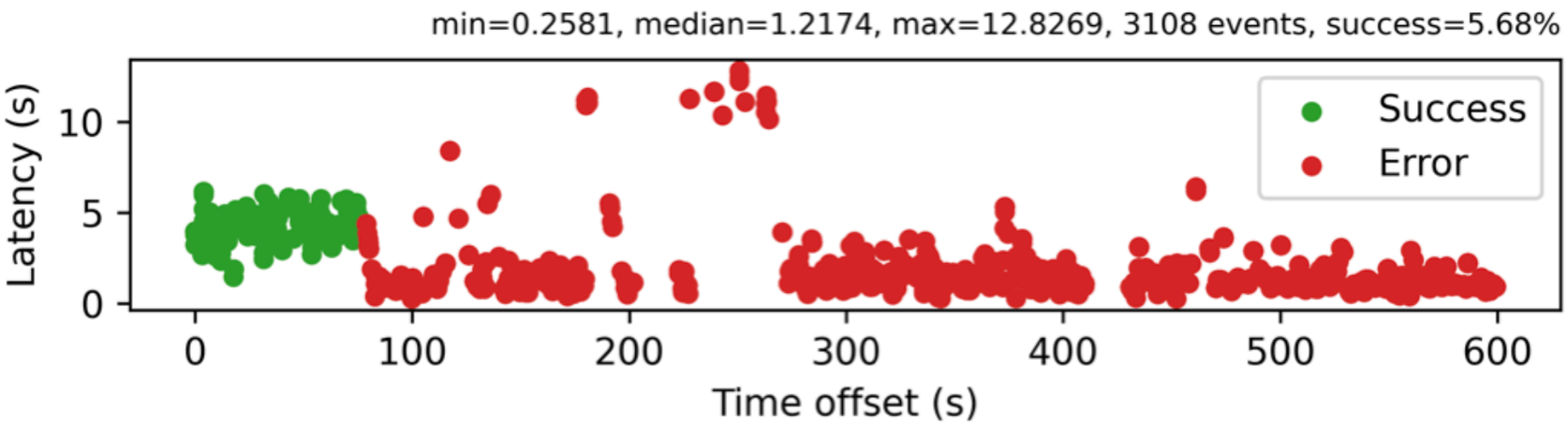
(a) Dyninka



(b) OpenFaaS

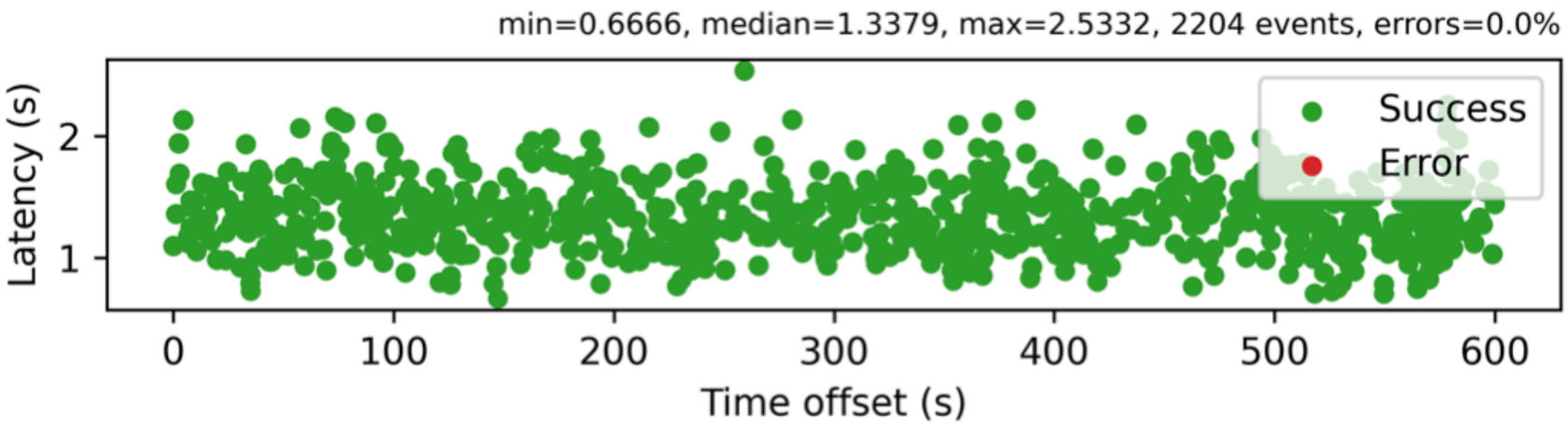
1 worker

Automatic Service Deployment



OpenFaas

10 workers



Dyninka

On-going projects

- Automatic service deployment in dynamic environments
- Service re-orchestration
- Service virtualization in automotive applications

Future works?

SPIE

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

citi lab

Edge AI: Data Flow Analysis in Edge Infrastructures

Chair INSA Lyon - SPIE ICS

Frédéric Le Mouël (INSA Lyon)

UNIVERSITÉ DE LYON

AI & Data Flow

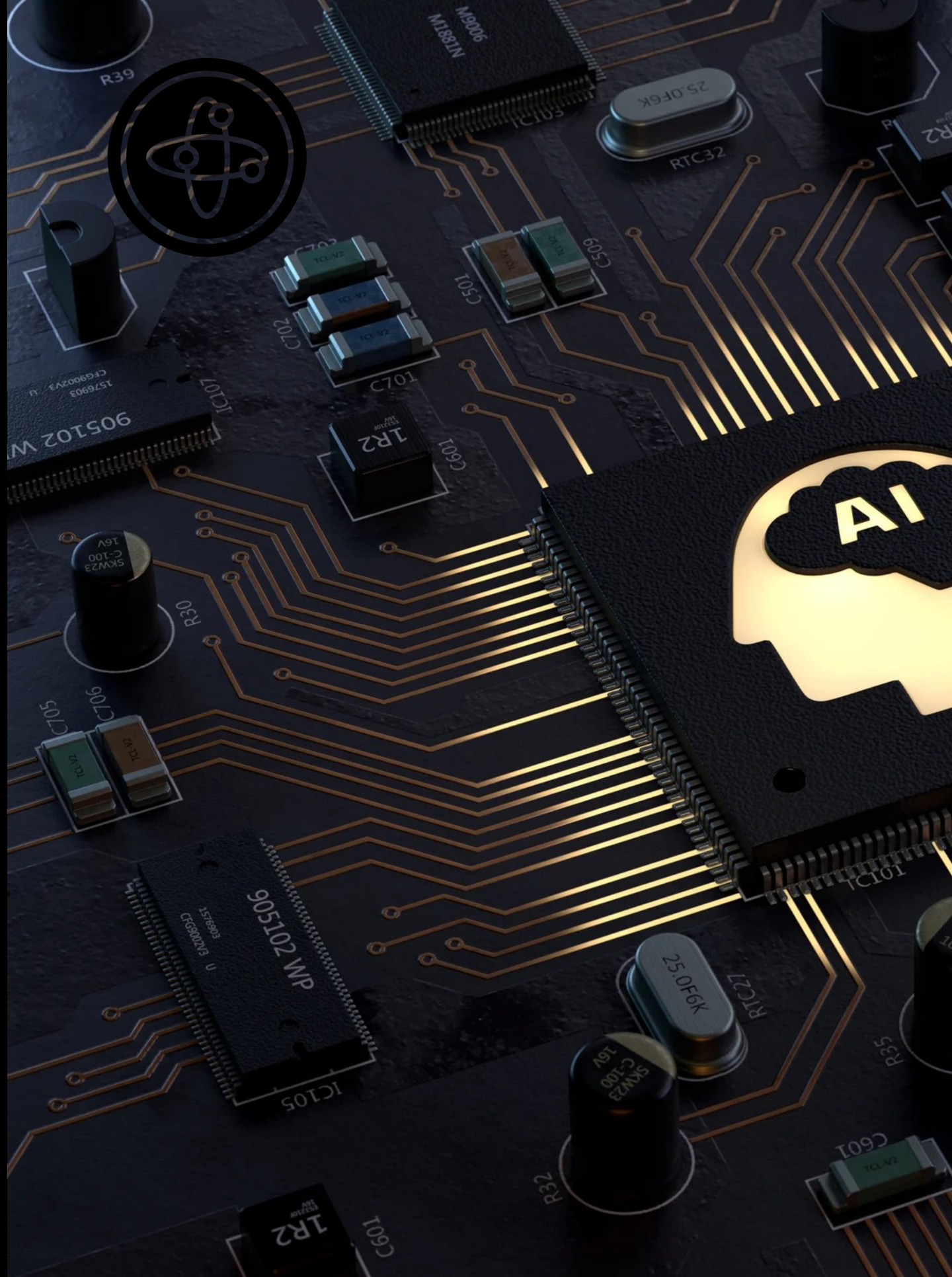
AI & Data Flow

- Sensing as a Network Service
- Network-traffic patterns in CDN
- Dynamic, Adaptive, Mobile, Multi-Tenant Slicing for E2E QoS

AI & Data Flow

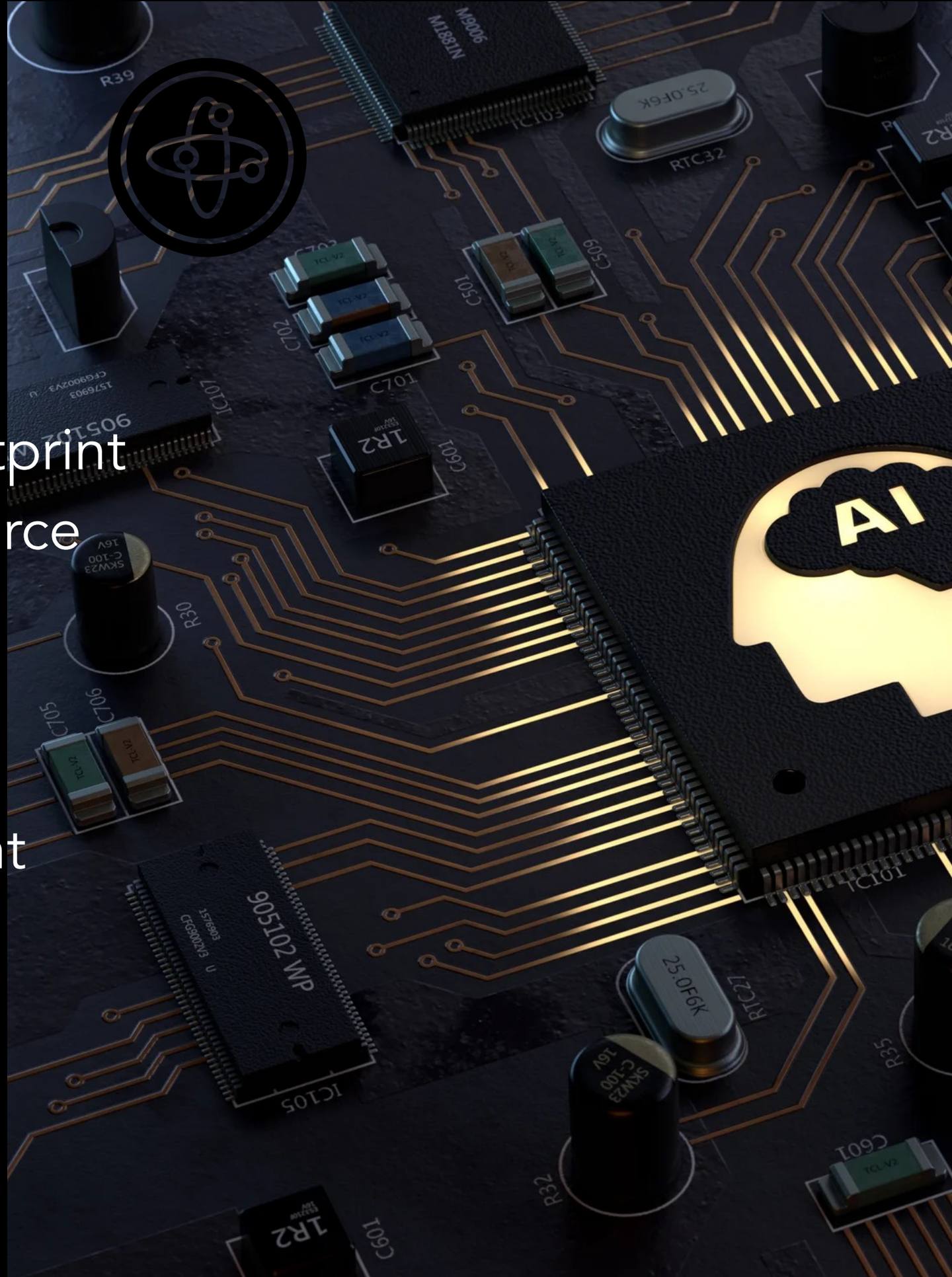
- Sensing as a Network Service
- Network-traffic patterns in CDN
- Dynamic, Adaptive, Mobile, Multi-Tenant Slicing for E2E QoS

Embedded AI



Embedded AI

- Low carbon & energy-footprint AI - Learning under Resource Constraints
- Decision-Making under Uncertainty
- Intermittent Reinforcement Learning



Embedded AI

- Low carbon & energy-footprint AI - Learning under Resource Constraints
- Decision-Making under Uncertainty
- Intermittent Reinforcement Learning

Geo-distributed AI

Geo-distributed AI

- Service Handover in Edge/Fog Computing Infrastructures for ML applications
- Low-Energy Federated Learning
- Fault-tolerant CNN & Generative Adversarial Learning

Geo-distributed AI

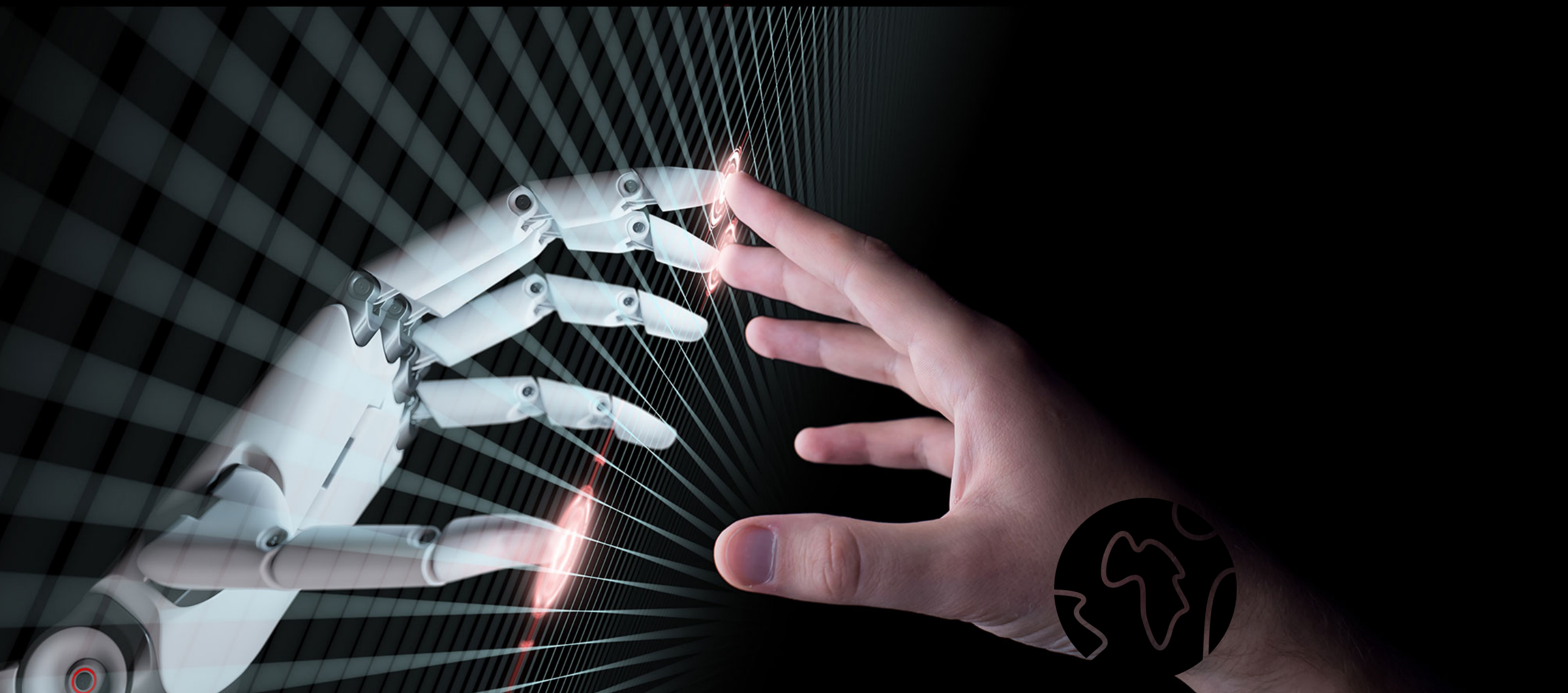
- Service Handover in Edge/Fog Computing Infrastructures for ML applications
- Low-Energy Federated Learning
- Fault-tolerant CNN & Generative Adversarial Learning

Cheap Local-IaaS

Self-Managed Infrastructures

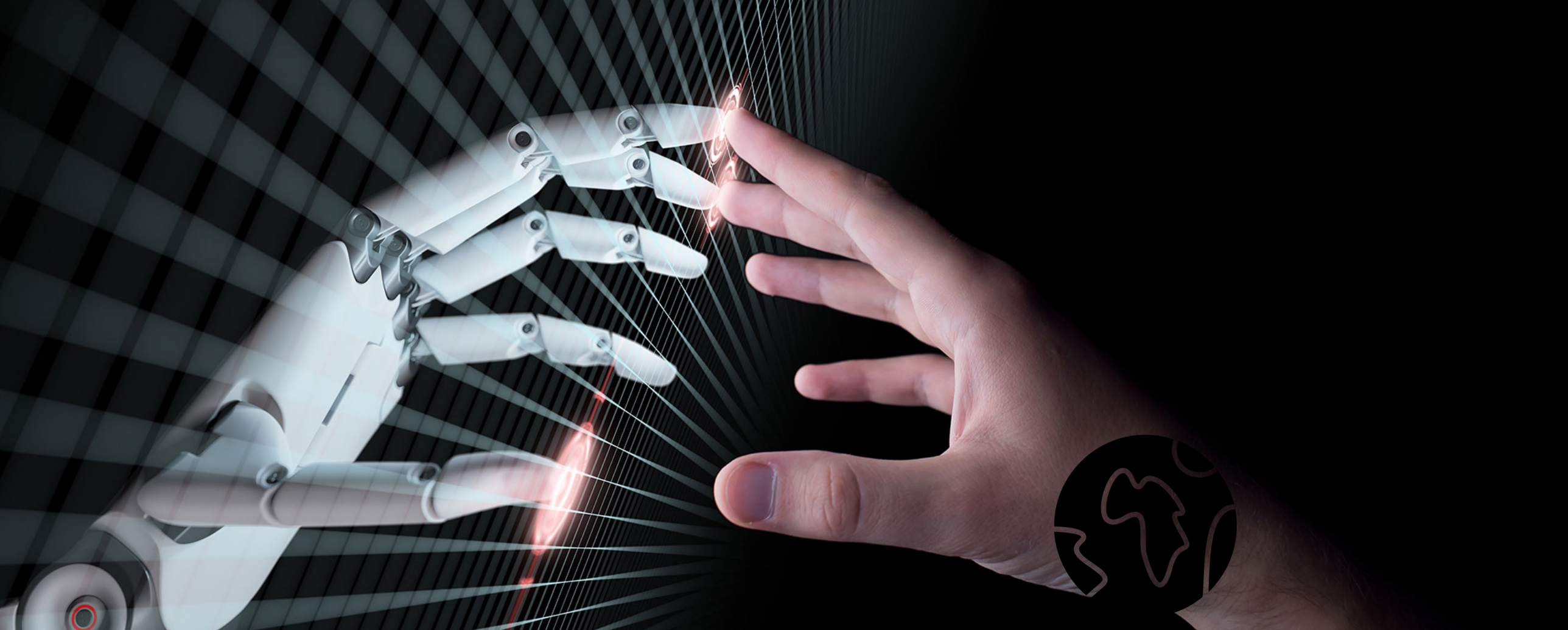
Remote Continuous Audits

Responsibility & Data Control



Responsibility & Data Control

- Trust & Privacy
- Sovereignty
- Embedded Ethics



Responsibility & Data Control

- Trust & Privacy
- Sovereignty
- Embedded Ethics

CITI Laboratory
http://www.citi-lab.fr

Head: Frédéric Le Mouël (frederic.le-mouel@insa-lyon.fr)
@flemouel