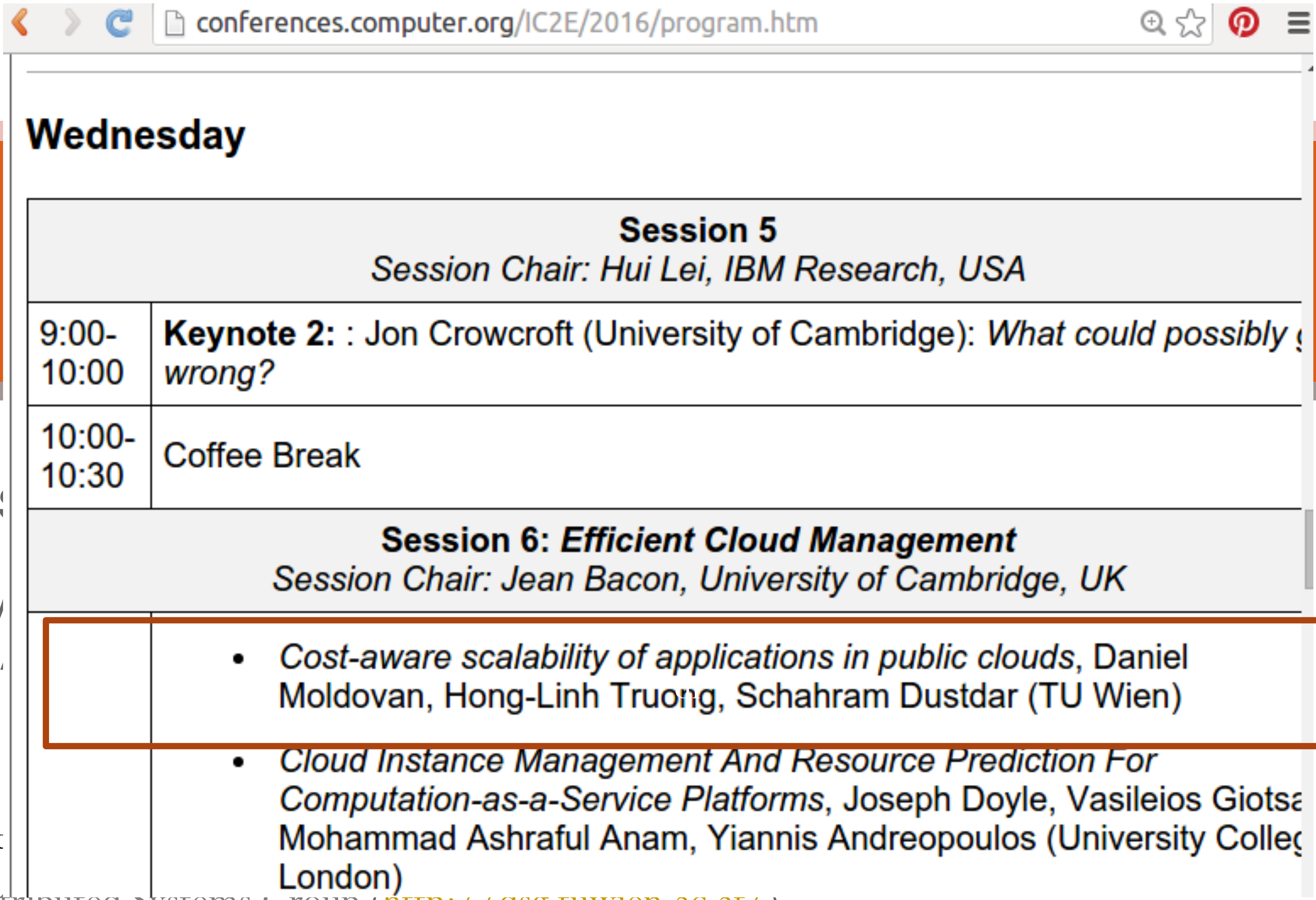


conferences.computer.org/IC2E/2016/program.htm

Wednesday

Session 5 *Session Chair: Hui Lei, IBM Research, USA*	
9:00-10:00	**Keynote 2:** : Jon Crowcroft (University of Cambridge): *What could possibly wrong?*
10:00-10:30	Coffee Break
Session 6: *Efficient Cloud Management* *Session Chair: Jean Bacon, University of Cambridge, UK*	
	• *Cost-aware scalability of applications in public clouds*, Daniel Moldovan, Hong-Linh Truong, Schahram Dustdar (TU Wien) • *Cloud Instance Management And Resource Prediction For Computation-as-a-Service Platforms*, Joseph Doyle, Vasileios Giotsa Mohammad Ashraful Anam, Yiannis Andreopoulos (University Colleg London)

Ay

Disc

Distributed Systems Group (http://dsg.tuwien.ac.at/)

Vienna University of Technology (http://www.tuwien.ac.at/)

Outline

- Introduction
  - Motivation
  - Context
- Patterns
  - Edge Provisioning Pattern
  - Edge Code Deployment Pipeline
  - Edge Orchestration Pattern
  - Diameter of Things (DoT)
- Conclusions

Building IoT Applications in Constrained Environments

- **Things:** Uniquely identifiable nodes using IP connectivity e.g., sensors, devices.
- **Gateways:** act as intermediaries between things and the cloud to provide the needed Internet connectivity, security and manageability.
- **Constrained-Node:** Tight limits on power, memory, and processing resources
- **Constrained-Node Networks:** Low throughput, high packet loss and limits on reachability over time
- **Cloud Infrastructure:** Large pools of virtualized servers and storage

Motivation / IoT Patterns Objectives

- Things in IoT are diverse:
  - In protocols
  - Capabilities
  - Computational power, etc.

- A well-designed IoT application:
  - Utilized edge devices
  - Fine-grained micro-services
  - Gateways to connect to Internet
  - Mobile/web Apps to interact with the things

- Designing for IoT is complex

Towards IoT Patterns

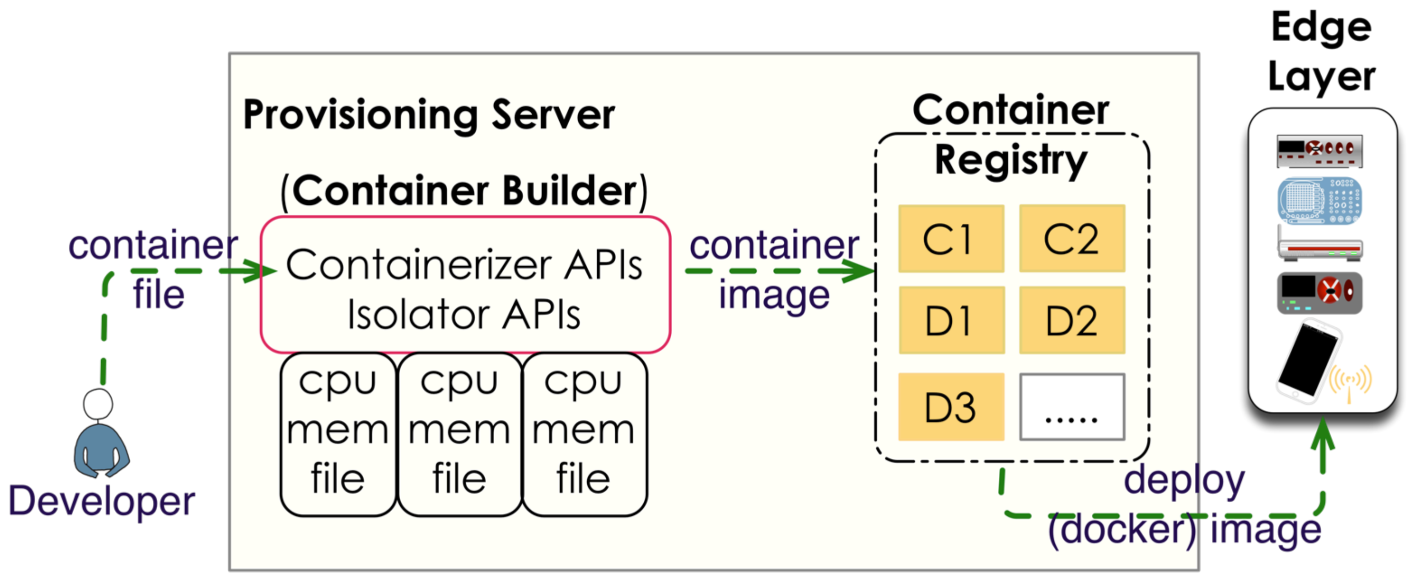
- We propose a set of reusable and abstract solutions to aid system architects in modeling and building IoT applications
- Abstract prescriptive design principles of frequent requirements tailored for IoT applications
- To create valuable, appealing, usable, and coherent edge applications

Edge Provisioning Pattern (1/2)

- **Context:**
  - IoT devices are usually geographically distributed,
  - Geographically distributed devices are hard to manage.
  - We need to reconfigure devices or provision new ones in an efficient way
- **Motivation:**
  - Suppose you have designed a system to display billboard.
  - At some point, you need to replace your technology stack entirely and provision a new environment remotely.
  - You may also want to add new devices and provision their runtime environment and applications quickly.
- **Problem:**
  - How can operation managers and developers ensure all of their edge devices are started with a reliable baseline environment, as needed?
  - How can they provision all the devices automatically all at once?

Edge Provisioning Pattern (2/2)

- **Container-based virtualization:**
  - Containers are good choice for provisioning resources, as they contain not only the code but also all other software dependencies, configurations and the whole runtime environment.
    - Container images utilize a layered and versioned file system, which has three benefits:
      - The devices can only pull the layers they need and not the whole image.
      - They can rollback to the latest or any working version of the image.

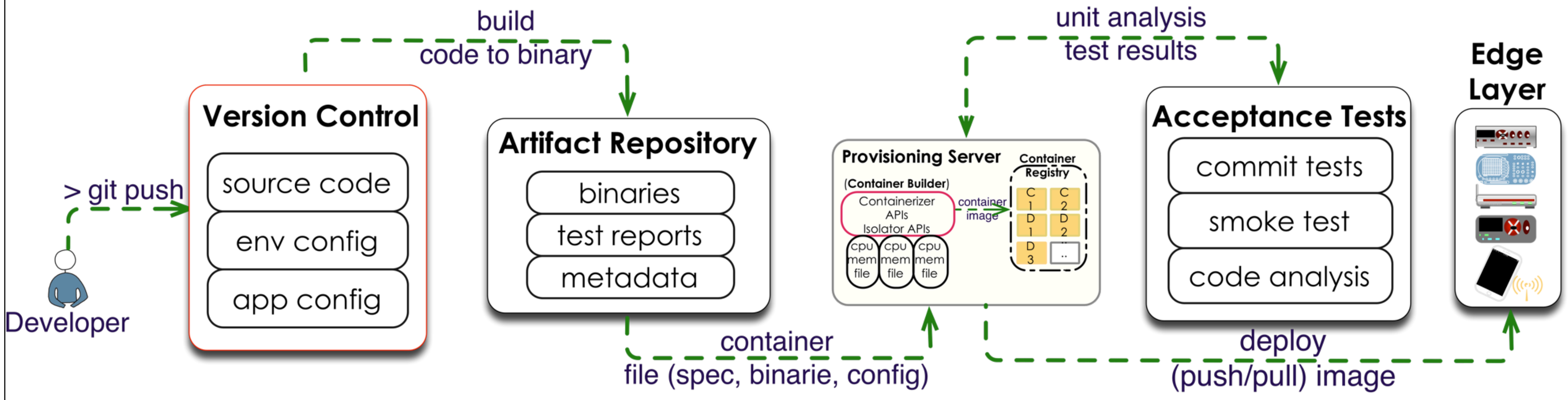


Edge Code Deployment Pattern (1/2)

- **Context:**
  - Maintainability is a main factor while deploying code to remote IoT devices.
  - As developers enhance and improve the code or fix some critical bugs, they expect to deploy the updated code to their several remote IoT devices quickly.
- **Motivation:**
  - Suppose you have designed a system to display advertisements on some billboards widespread in a region.
  - You need to update the text or graphical features frequently or change the duration of ad display.
  - Maintainability and adaptability are the most important challenges in such designs. You must be able to update the code and deploy it to all your devices at once.
- **Problem:**
  - How can developers deploy their code to many IoT devices automatically, quickly and safely, and configure them without being concerned about the long process of build, test and release?

Edge Code Deployment Pattern (2/2)

- **Device Continuous Delivery Pipeline:**
  - Mitigating connectivity issues, it is best to only deliver changes and not the application as a whole across the constrained network.
  - The pipeline includes building the application, its deployment and testing and finally releasing and distributing it to edge devices.
  - Devices can periodically ask the central registry/hub for new versions or the server can notify devices about a release of a new image version

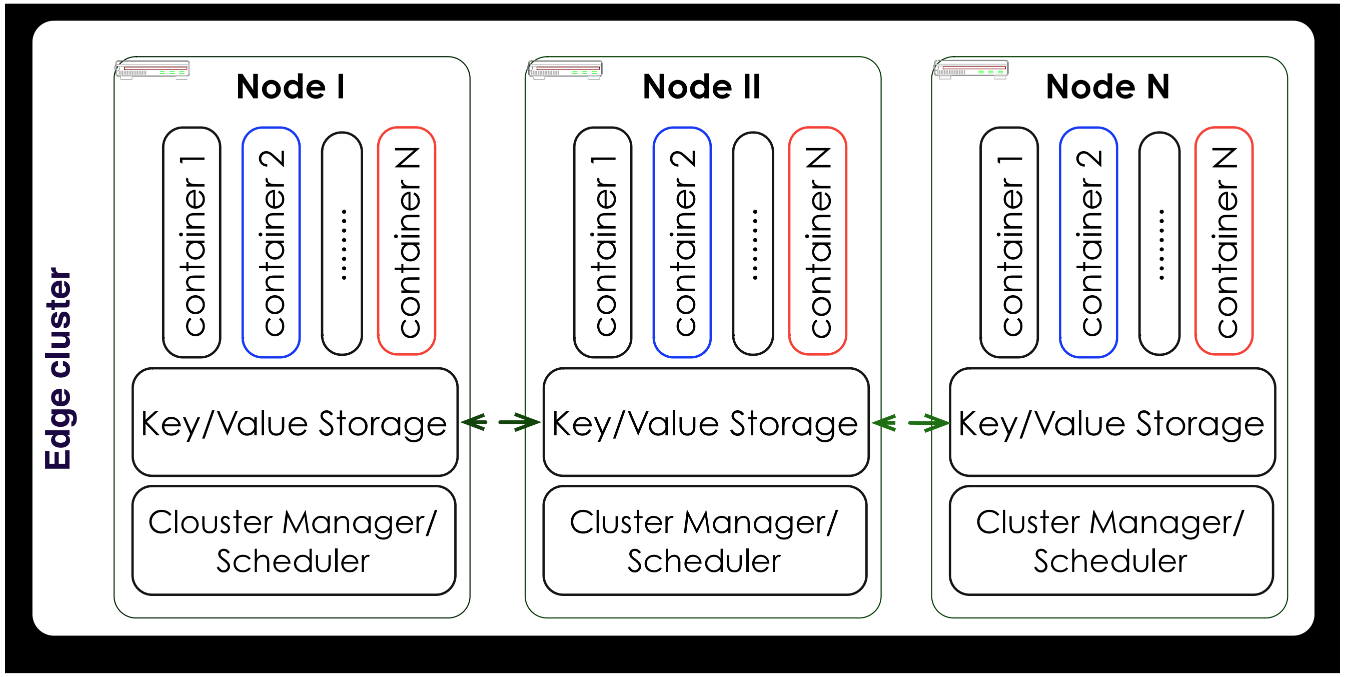


Edge Orchestration Pattern (1/2)

- **Context:**
  - Enabling a large number of devices connected via edge means empowering the cluster to manage its nodes to check their health state, their services state to reconfigure them.
  - Moreover, in case of IoT devices to adapt and to calibrate nodes remotely and quickly.
- **Motivation:**
  - Suppose you have designed a system to display advertisements on some billboards widespread in a region. You have devices to control each billboard. You want to be able to check their state and health status, manage them, check their services, change their runtime configuration, and execute services in the cluster or on certain devices.
- **Problem:**
  - How can we orchestrate IoT devices in accordance with their tightly scripted configurations as nodes of a cluster remotely?
  - How can edge cluster nodes discover services?

Edge Orchestration Pattern (2/2)

- **Automated IoT Edge Cluster Management**
  - The architecture should avoid having a single point of failure.
  - Nodes are able to advertise their roles and services, as well as to discover each other.
  - Each node can be the manager, hence on its failure a new manager will be elected in the cluster.

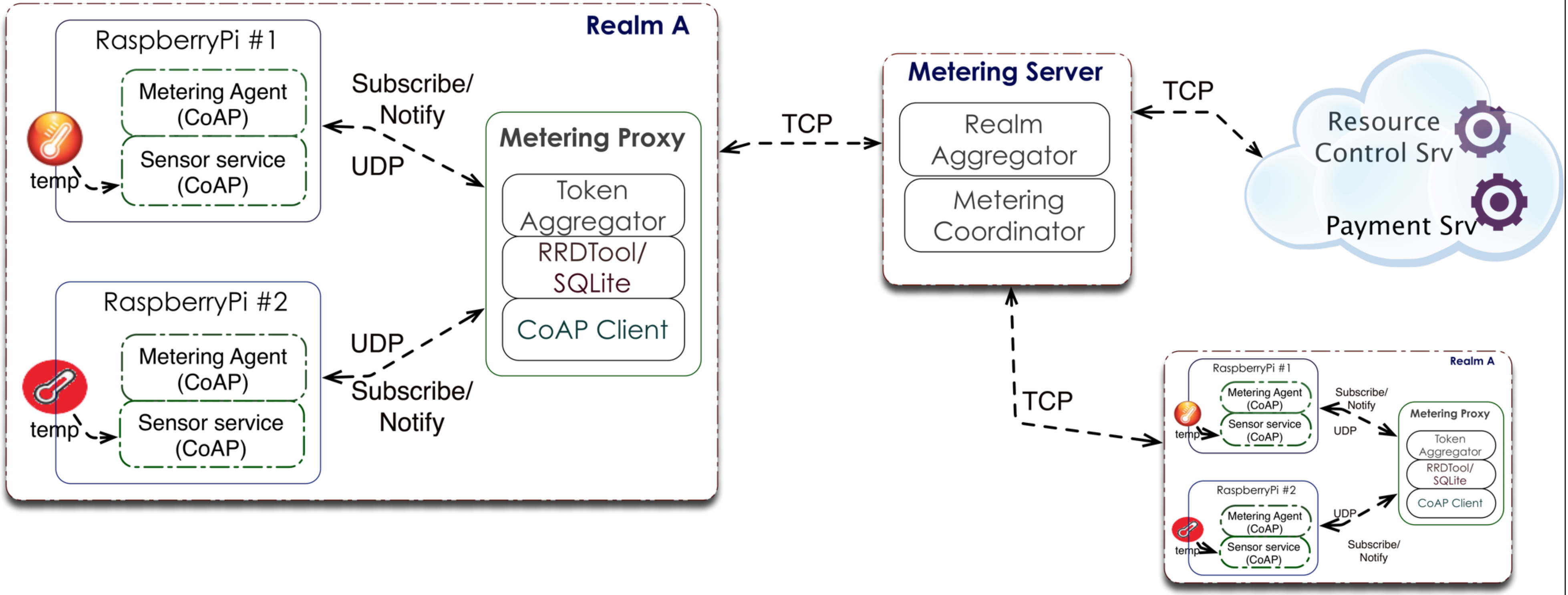


Diameter of Things (DoT) Pattern (1/2)

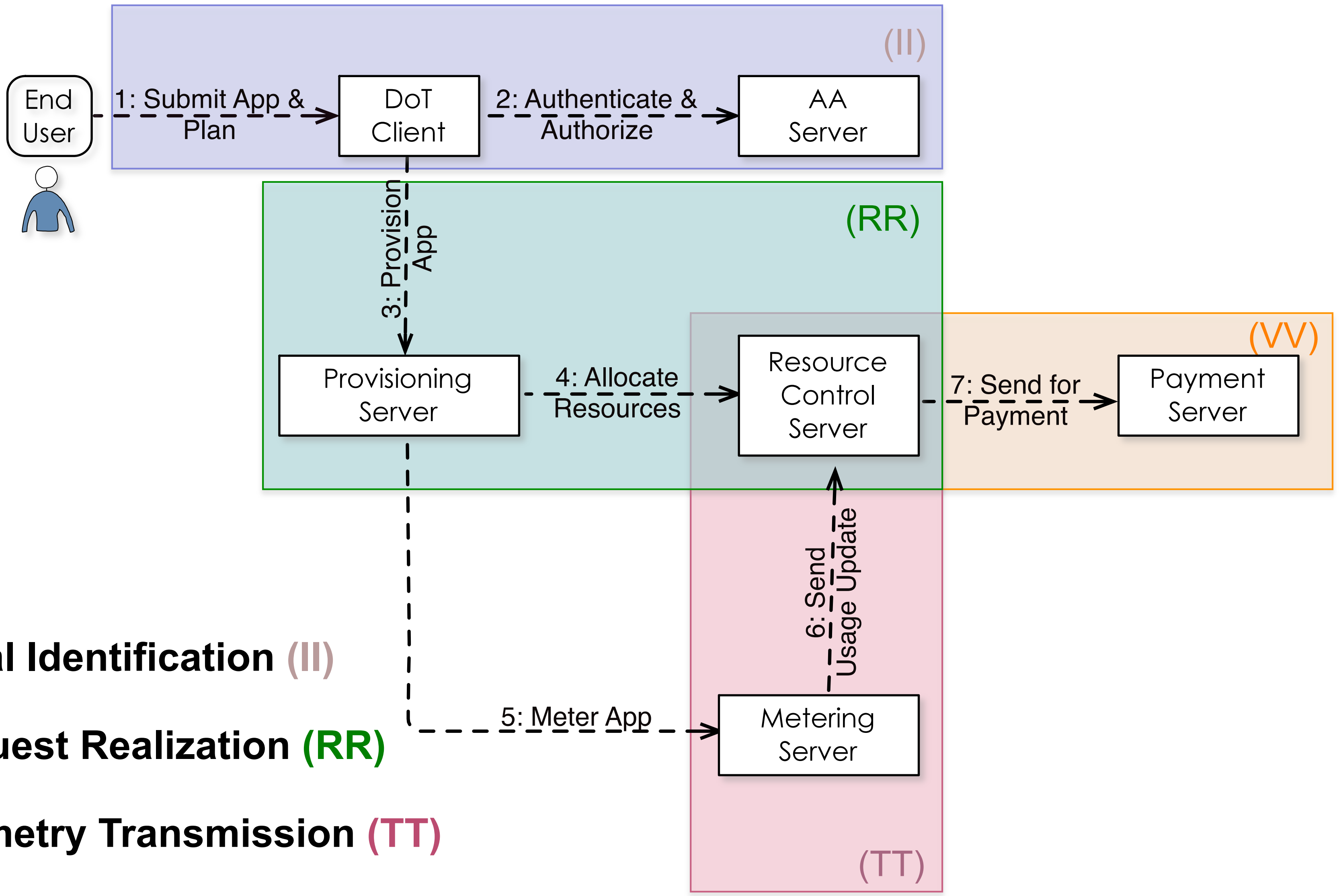
- **Context:**
  - From the provider's perspective metering mechanisms can vary based on applied business models
  - These mechanisms range from invocation basis (event-based) and usage over time (time-based), to subscription models such as prepaid or pay-per-use
- **Motivation:**
  - You have provided an IoT platform consisting of devices and composite IoT services and you would like to monetize it based on the real usage of the client, while guaranteeing fairness.
- **Problem:**
  - How can IoT service provider monitor and meter the actual usage of IoT deployment units in real-time or near-real-time, in order to monetize them?
  - How the IoT composite application resource usage, as well as the service usage can be charged against a specific user balance?
  - How to collect information on resource usage for the purpose of capacity planning, auditing, billing, or credit allocation?

Diameter of Things (DoT) Pattern (2/2)

- **Metering Patterns:**
  - **Timers:** Express duration of a particular method call.
  - **Counters:** Measure number of calls, amount of data.
  - **Gauges:** Represent a single variable changing over the time .
- **Metering Framework:**



Diameter of Things (DoT) Framework



- **Initial Identification (II)**
- **Request Realization (RR)**
- **Telemetry Transmission (TT)**
- **Value Verification (VV)**

Conclusion (1/2)

- **Edge Provisioning Pattern:** How can operation managers and developers ensure all of their edge devices are started with a reliable baseline environment, as needed?
  - Bundling software dependencies, configurations and the whole runtime environment into containers.
  - Layered provisioning of images contributes to an more utilized versioned system.

- **Edge Code Deployment Pattern:** How can operation managers and developers ensure all of their edge devices are started with a reliable baseline environment, as needed?
  - Transparent code deployment to the devices by the developers
  - Deliver the changes and not the application as a whole across the constrained network

Conclusion (2/2)

- **Edge Orchestration Pattern:** How can we orchestrate IoT devices in accordance with their tightly scripted configurations as nodes of a cluster remotely?
  - Each node must be able to know the state of the whole cluster.
  - Via device pairing, devices trust each other and start sharing data or trigger functionality.
  - Node manager election on failure.

- **DoT Metering Pattern:** How to dynamically meter and price IoT applications? What are the usage patterns?
  - Real-time telemetry of IoT applications
  - Event-based (invocation-basis) & Time-based (usage over time) Metering patterns
  - Subscription models such as prepaid and pay-per-use models

Thank you

Dr. Soheil Qanbari:

- E-mail: soheil@dsg.tuwien.ac.at
- Website: http://www.infosys.tuwien.ac.at/staff/soheil/

Distributed Systems Group (http://dsg.tuwien.ac.at/)

Vienna University of Technology (http://www.tuwien.ac.at/)