Managing and Testing Ensembles of IoT, Network functions, and Clouds

Hong-Linh Truong

Distributed Systems Group, TU Wien
truong@dsg.tuwien.ac.at
http://rdsea.github.io
Acknowledgements:

Many results are from the joint work with students and colleagues:
Luca Berardinelli, Duc-Hung Le, Nanjangud C. Narendra, Christian Proinger, Daniel Moldovan, Ivan Pavkovic, Georgiana Copil, Stefan Nastic

and from

H2020 U-test (www.u-test.eu)
H2020 Inter-IoT (http://www.inter-iot-project.eu/)

Note: some work are under submission.
Outline

- Application-oriented ensembles of IoT, Network functions and Cloud resources
- Resource Management
- Monitoring
- Uncertainty and Testing Uncertainties of Infrastructures
- Conclusions and Future work
Application-oriented Ensembles of IoT, Network functions, & Clouds: Necessity?
Systems of IoT, Network Functions, and Clouds

Management layer

Manage and provision resource slices

Virtual resource layer

deployed on edge devices

Physical resource layer

based on network function virtualization and software-defined network

running on multiple cloud data centers

CCI@USC, Los Angeles, 21 Sep 2017
Moving to blending IoT, network functions and cloud resources

- **Type 1**
  - **Mainly focus** on IoT networks: sensors, IoT gateways, IoT-to-cloud connectivity (e.g., connect to predix.io, IBM Bluemix, Azure IoT, Google Cloud, Amazon IoT, etc.)

- **Type 2**
  - **Mainly focus** on (public/private) services in data centers: e.g., IoT data hubs, NoSQL databases, and big data ingest systems

- **Type 3**
  - **Equally focus** on both IoT and cloud sides and have the need to control at both sides
  - Highly interactions between the two sides, including the network in the middle
All types of service models

- Cloud resources as services are known
- NFV/5G coming and more service providers at the edge are available
  - Network function virtualization
- IoT infrastructure-as-a-service
  - Pay-per-use IoT communication
- IoT data-as-a-service Cloud service models
  - Public and private providers
Examples of IoT networks

Enabled by Semtech’s LoRa technology, which allows a sensor network to operate on low power while providing strong connectivity over a long distance, the SK Telecom LPWAN covers 99 percent of South Korea’s population. The company predicts that it will have over four million things connected to its IoT networks by the end of 2017.

The LoRaWAN™ technology platform based on Semtech’s LoRa two-way wireless RF technology is a comprehensive solution for low power, long-range connectivity, which is essential for LPWANs supporting IoT applications. It offers deep penetrability, secure connectivity, long battery lifetime and streamlined implementation for simple network rollout, as well as integration into existing infrastructure.

Some application scenarios

- Emergency responses, on-demand crowd sensing, Geo Sports monitoring, cyber-physical systems testing, etc.

Geo Sports: Picture courtesy Future Position X, Sweden
Emergency in a Seaport

- Built a top existing INTER-IoT scenarios
  - Coordinating activities when an accident happens
- Example of actions in a resource slice in a seaport:
  1) activating monitoring containers with sensitive goods in the port
  2) analyzing and controlling robotic cranes and trucks to make sure that they do not prevent the emergency responses as well as ready to support the responses
  3) sending alarms and controlling vessel arrivals and revising transport planning,
  4) providing information for operational assistance for the emergency responses
  5) activating systems to support the monitoring of people impacted by the accident using devices and platforms for chronic disease and cognitive decline prevention.
Examples of IoT data services

IoT data as a service can be offered by different types of providers

Ensembles of IoT, Network functions and clouds

Functional Requirement (e.g., data) → Performance requirements → Uncertainty requirements → Cost requirement → Security requirements

IoT Application
- Application
- Application
- Application

Ensemble of IoT Services and Network Services (at the edge)

utilizes and adapts ensembles

provides capabilities

Service Capability Abstraction
- IoT services
- Network services

abstract and present service capabilities

IoT Networks

API
- Gateway
- Virtual Sensor
- Deployment
- Actuator
- Sensor

Clouds

API
- Loadbalancer
- Data Analytics
- Cloud storage
- User Management

Network Functions Services
- Routing
- Naming
- NFV Orchestration

CCI@USC, Los Angeles, 21 Sep 2017
Challenges

- **Modeling** distributed IoT, network functions and cloud capabilities in an ensemble
- **Slicing** end-to-end network of resources
- **Composing** resources in ensembles of IoT, network functions and clouds
- **(Re-)configuring** composed resources
- **Testing and monitoring**

The SINConcept: http://sincconcept.github.io/
Tools treating IoT, network functions, and cloud services in an isolated manner are not enough
Service engineering analytics for IoT Cloud Systems

We started in 2011 without network functions in our mind!

http://tuwiendsg.github.io/iCOMOT/
Monitoring, Controlling and Testing IoT Cloud Systems

Lightweight analytics and control → IoT cloud applications → Large-scale data analytics

Sensors → Gateways
Edge: IoT units

IoT cloud systems: Software layer

Sensor data → Load balancer → Event Handling Web service → Message-oriented middleware → Near-real-time data processing
Cloud: Cloud services

NoSQL big data

Deploy, configure, govern, and control

IoT governance (GovOps) → Deployment and configuration (Salsa) → Monitoring and analytics (Mela)

iCOMOT services and tools

Check: http://tuwiendsg.github.io/iCOMOT/demo.html
Configuration is not done easy for the ensembles

- Sensors/gateways and cloud services have different management systems and interfaces
  - Deploy different topologies and configure topologies to work together is hard
Cross IoT and cloud controls

- IoT gateways control is very different from cloud controls
- Low-level REST API in IoT gateways management and high level elasticity rules for cloud services
- Different protocols for communicating with resource managers
Too low-level Infrastructure-level Testing

System Static Structure Description

Run-time System Structure Management

Elastic System

Added
Component X

Removed
Component Y

Verification Strategies Definition

What to verify?
When to verify?
How to verify?

APIs / Metrics
Component X

APIs / Metrics
Component Y

Verification Strategies Enforcement

Verify X Directly

Verify X Indirectly

Verification Result Events

Notify

Software Controller

Human

*testing strategy=testing plan


CCI@USC, Los Angeles, 21 Sep 2017
Duc-Hung Le, Nanjiangud C. Narendra, Hong Linh Truong:

HARMONIZING RESOURCES FOR ENSEMBLES?
Integrating diverse types of resources

- Make a Resource Grid ready for creating ensembles
  - Harmonize IoT, network functions and cloud resources
- API Integration and Communication
  - Use REST API for obtaining metadata and control of resources
    - Sensoring data can be transferred through different middleware
  - Work with existing metamodel (IoTivity, OpenHAB, IoTDM, ETSI MANO, OCCI, CIMI, etc.)
- Rely on scalable cloud communication middleware (e.g., AMQP & MQTT)
Examples of existing providers/models

<table>
<thead>
<tr>
<th>Provider</th>
<th>Category</th>
<th>APIs</th>
<th>Information models</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIWare Orion</td>
<td>IoT</td>
<td>RESTful (NGSI10), one-time query or subscription</td>
<td>High level attributes on data and context</td>
</tr>
<tr>
<td>FIWare IDAS</td>
<td>IoT</td>
<td>RESTful for read/write custom models and assets</td>
<td>Low level resource model catalogs</td>
</tr>
<tr>
<td>IoTivity</td>
<td>IoT</td>
<td>REST-like OIC protocol, support C++, Java and JavaScript</td>
<td>Multiple OIC model</td>
</tr>
<tr>
<td>OpenHAB</td>
<td>IoT</td>
<td>RESTful for query and control IoT resources</td>
<td>Low level resource model catalogs</td>
</tr>
<tr>
<td>OpenDayLight</td>
<td>Network</td>
<td>Dynamic REST generated from Yang model (model-driven)</td>
<td>Low level resource model catalogs</td>
</tr>
<tr>
<td>OpenBaton</td>
<td>Network</td>
<td>RESTful for network service description</td>
<td>ETSI MANO v1.1.1 data model</td>
</tr>
<tr>
<td>OpenStack</td>
<td>Cloud</td>
<td>RESTful, multiple language via SDK, OCCI, CIMI</td>
<td>OpenStack model, OCCI, CIMI</td>
</tr>
</tbody>
</table>
Information model

**Physical:** Sensor/actuator/devices in providers’ models

**Virtual IoT:** SD-Gateway and capabilities.

**Network functions:** edge-to-edge, edge-to-cloud network.

**Clouds:** VM, data services, data analytics.

CCI@USC, Los Angeles, 21 Sep 2017
Prototype

**HINC** Harmonizing IoT, Network functions and Clouds

**Brief**

HINC implements the lowest layer of the SINC Concept by interfacing to different providers to obtain resource metadata and to control resources. HINC concepts and techniques are described in the paper: Duc-Hung Le, Nandagud Narendra, Hong-Linh Truong HINC: Harmonizing Diverse Resource Information Across IoT, Network Functions and Clouds (Submitted PDF), The IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud2016), 22-24 Aug, 2016, Vienna, Austria

**Scope**

HINC framework provides an abstraction layer on top of IoT cloud infrastructure for the IoT cloud application. To this end, HINC focuses on distributed architecture and communication protocol to enable the interoperation of IoT services on both edge-to-edge and edge-to-cloud networks. HINC brings network functions into the picture that aims to enable the reconfiguration of the network and communication capability among IoT services and cloud services.

![Diagram of HINC framework and IoT cloud infrastructure](http://sincconcept.github.io/HINC/)

**Highlight features**

1. High level view of distributed resources for IoT, Network functions and Clouds.

http://sincconcept.github.io/HINC/

CCI@USC, Los Angeles, 21 Sep 2017
Testbed

Virtual IoT resource
- Docker
- Sensors
- Database (OrientedDB)

Virtual router (docker)
- Has capability
- Connect/disconnect with other routers

Virtual Network Functions
- Virtual router (docker)
- Virtual router (docker)
- RabbitMQ (LAN)

Cloud services
- Virtual router (docker)
- Event Processing (docker)

In-lab testbed

Local Management Service
- Capability

RabbitMQ
- www.cloudamqp.com

Global Management Service
- ec2.amazon.com

OpenHAB provider
- Pull data

Virtual IoT resource - India

Virtual IoT resource - Austria

Network functions - India

CCI@USC, Los Angeles, 21 Sep 2017
Reducing complexity in accessing and control resources

1. Query data points

```java
// specify the sensors whose sample rate needs to be changed
DataPoint template = new DataPoint('BodyTemperature');
SensorProps sensorProps = new SensorProps();
sensorProps.setRate(5);
template.getExtra().add(new PhysicalResource(sensorProps));
QueryManager queryMng = new QueryManager('ampq://10.99..');
List datapoints = queryMng.QueryDataPoints(template);

// observe the resource and send back the control
for (DataPoint dp : datapoints) {
    DataPointObserver obs = new DataPointObserver(dp) {
        public void onChange(DataPoint newVal) {
            SensorProps props = newVal.getExtra('SensorProps');
            if (props.getRate() > 5) {
                props.setRate(5);
            }
            ControlPoint control = newVal.getControl('changeRate', 5);
            queryMng.SendControl(control);
        }
    }
}
```

2. Control the resource

```java
// create the template of the query
NetworkFunctionService nfsTemp = new NetworkFunctionService(nfsTemp.getQuality().setBandwidth("16 GB/s");
CloudService cloudTemplate = new CloudService('storage');
cloudTemplate.hasAttribute("capacity", "1 TB");

// Query the list of resources
QueryManager queryMng = new QueryManager('ampq://10.99..');
List networks = queryMng.QueryNetworkFunctionService(nfsTemp
List storages = queryMng.QueryCloudService(cloudTemplate);
// further queries
```

3. Query network functions and clouds
As we able to get resources $\rightarrow$ deploy and configuration
MONITORING ENSEMBLES?
Key questions in monitoring

- Which metrics make sense? Memory/CPU usage?
  - We focus on: response time, delay, cost, availability and reliability
- Collect monitoring data from multiple resources of sub-systems
  - Gathering and correlating monitoring
  - Deal with different integration models: through provider interfaces versus instrumentation
- Analytics and visualization
Monitoring components

Example of an ensemble
Example of low level metrics
UNCERTAINTY AND APPROACH TO TESTING
Infrastructure uncertainties

- CPS: includes IoT, network functions and clouds
- An ensemble represents a virtual infrastructure
  - Cross-system and cross-layer
- Goals
  - Characterize and specify possible uncertainties associated with interactions among elements in CPS infrastructures
- Supporting testing uncertainties and uncertainties analytics
  - Conventional aspects, e.g., infrastructural physical resources and typical system operations
  - Emerging novel aspects: data uncertainties (data/data-centric CPS), elasticity of CPS resources (w.r.t function and composition), and Governance (related to business/trustworthiness)
Uncertainty Model

Characterizing Uncertainty

Belief Model

Source: https://www.simula.no/file/d12pdf/download
Important infrastructure uncertainties classes

Data

Governance

Elasticity
Examples – UC1 Geo Sports

Data Delivery uncertainties
Uncertainties affect the infrastructure capabilities for generating, processing and delivering data

Located in Software in Cyber Environment, e.g., software components (execution environments, OS of gateways) in the infrastructure

At Runtime but Sporadic
Caused by Resource Competition (Resource Competition)

Indirect impact on Software

Risk of this type of uncertainties

<table>
<thead>
<tr>
<th>Non-functional aspects</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Extreme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Legal/compliance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Combining MDE and Elastic Execution

Key observations:

how to combine topdown approach and bottom-up approach to leverage strengths of both models and elastic systems?
Key issues: Two separate worlds

Preparing CPS under Test

Figure source:
Combining MDE with Elastic Execution

Interwoven test execution and provisioning

- Uncertainty Profile
  - Modeling Uncertainties and SUT
    - System Under Test (SUT) Models
      - Deploying/Configuring SUT
        - IoT/Cloud Resource Information
  - Generating Uncertainty Test cases
    - Deploying/Configuring Testing Utilities
      - Executing Tests
        - Test cases
          - System Under Test (SUT)
            - Test Utilities
              - Deploying/Configuring SUT

CCI@USC, Los Angeles, 21 Sep 2017
Tool pipelines: from MDE to elastic execution for testing

Key thoughts:
- Different algorithms to create suitable deployment configurations based on data uncertainties, cost, and time
- Interactions between testing and elasticity control of IoT and Cloud services

CCI@USC, Los Angeles, 21 Sep 2017
Our current progress

Consider to generate provisioning configurations from SUT models

IoT and Cloud Resource Profile
Uncertainty Profile
Generic models for Task Executor

Tasks for testing also including deployment and reconfiguration of ensembles.
Configuration Generation and Deployment

- Reuse well-known tools for deployment
- Utilize different information services
  - Runtime and (future) uncertainty analytics
Example of BTS monitoring
Example of BTS monitoring

```
"BeliefAgent"
  Developers

"BeliefElement"
  MQTTUsage

{ prerequisites = MQTTUsage }

"BeliefElement"
  MessageDelivery

1

"DataDeliveryUncertainty"
  MQTT Data Delivery Uncertainty

{ cause_type = technological,
effect_propagation_type = application,
functional_dimensionality_type = data_delivery,
non_functional_dependability_type = dependability,
temporal_manifestation_type = recurring,
observation_type = deployment_time,
locality_type = platform,
indeterminacy_source = MQTTConfig1 }
```
Examples

Enriched model information for deployment configurations

```
"MQTTConfig1": {  
  "name": "MQTTConfigServer",  
  "protocolType": "MQTT",  
  "qosLevel": [],  
  "type": "CommunicationConfiguration"
}
"MQTTConfig2": {  
  "name": "MQTTConfigClient",  
  "protocolType": "MQTT",  
  "clientID": "",  
  "serverIP": "35.189.187.208",  
  "portNumber": 1883,  
  "topics": ["/gateway/electricity"],  
  "qosLevel": [2],  
  "type": "CommunicationConfiguration"
}
```

Generated deployment description

```
services:
  ingest:
    build: .
    volumes:
    - .:/t4u
electricitiesensor:
    image:
"localhost:5000/t4u/mqttsensor/realsensor: v01"
iotgateway:
    image:
"localhost:5000/t4u/cloudservice/mqttbroke
r:v01"
```
Summary

- Ensembles of IoT, network functions and clouds
  - Important for various types of applications
  - SINC: a conceptual framework

- Elasticity and Uncertainties
  - We need to consider more data and network aspects.
  - Adaptation and optimization under uncertainties
  - Modeling and testing uncertainties are very challenging
  - Testing uncertainties require fundamental changes in testing techniques: deal with elasticity and virtualization

→ We need new set of tools and techniques for managing and testing ensembles.
Future work

- **Current research topics**
  - APIs for programming resource queries and controls ([http://sincconcept.github.io/HINC/](http://sincconcept.github.io/HINC/))
  - Configuration tools ([http://tuwiendsg.github.io/SALSA/](http://tuwiendsg.github.io/SALSA/))
  - Uncertainty testing and analytics ([https://github.com/tuwiendsg/COMOT4U/](https://github.com/tuwiendsg/COMOT4U/))
  - Monitoring and analytics
  - Ensemble requirement modeling, composition algorithms and optimization
  - Interoperability issues ([http://www.inter-iot-project.eu/](http://www.inter-iot-project.eu/))
  - Policy execution

Check [http://rdsea.github.io](http://rdsea.github.io) and [https://github.com/tuwiendsg/COMOT4U](https://github.com/tuwiendsg/COMOT4U) for new update
Thanks for your attention!

Hong-Linh Truong
Distributed Systems Group
TU Wien
rdsea.github.io